尚硅谷大数据技术之Apache Paimon

（作者：尚硅谷研究院）

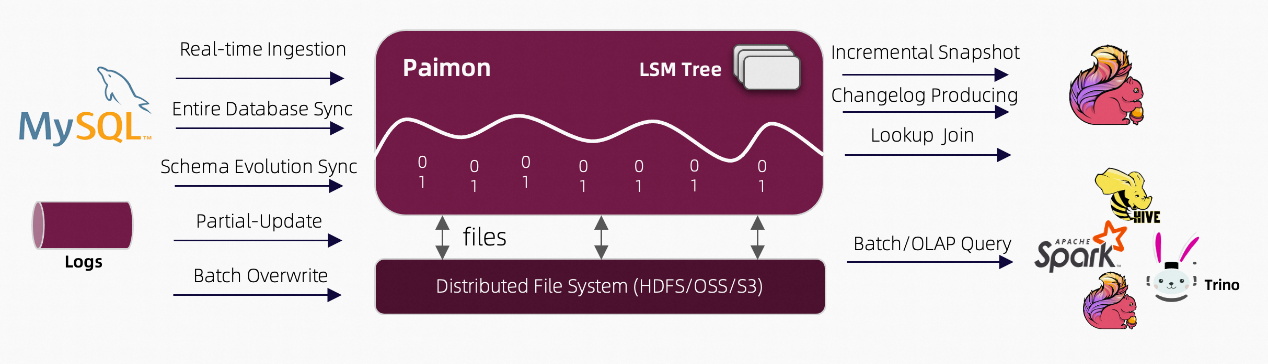
版本：V1.0

# 第1章 概述

## 1.1 简介

Flink 社区希望能够将 Flink 的 Streaming 实时计算能力和 Lakehouse 新架构优势进一步结合，推出新一代的 Streaming Lakehouse 技术，促进数据在数据湖上真正实时流动起来，并为用户提供实时离线一体化的开发体验。Flink 社区内部孵化了 Flink Table Store （简称 FTS ）子项目，一个真正面向 Streaming 以及 Realtime的数据湖存储项目。2023年3月12日，FTS进入 Apache 软件基金会 (ASF) 的孵化器，改名为 Apache Paimon (incubating)。

Apache Paimon是一个流数据湖平台，具有高速数据摄取、变更日志跟踪和高效的实时分析的能力。



**1）读/写：Paimon 支持多种读/写数据和执行 OLAP 查询的方式。**

（1）对于读取，它支持以下方式消费数据：

* 从历史快照（批处理模式），
* 从最新的偏移量（在流模式下），或
* 以混合方式读取增量快照。

（2）对于写入，它支持来自数据库变更日志（CDC）的流式同步或来自离线数据的批量插入/覆盖。

**2）生态系统**

除了Apache Flink之外，Paimon还支持Apache Hive、Apache Spark、Trino等其他计算引擎的读取。

**3）内部**

在底层，Paimon 将列式文件存储在文件系统/对象存储上，并使用 LSM 树结构来支持大量数据更新和高性能查询。

**4）统一存储**

对于 Apache Flink 这样的流引擎，通常有三种类型的连接器：

* 消息队列：例如 Apache Kafka，在源阶段和中间阶段都使用它，以保证延迟保持在秒级。
* OLAP系统：例如Clickhouse，它以流方式接收处理后的数据并为用户的即席查询提供服务。
* 批量存储：例如Apache Hive，它支持传统批处理的各种操作，包括INSERT OVERWRITE。

Paimon 提供表抽象。它的使用方式与传统数据库没有什么区别：

* 在批处理执行模式下，它就像一个Hive表，支持Batch SQL的各种操作。查询它以查看最新的快照。
* 在流执行模式下，它的作用就像一个消息队列。查询它的行为就像从历史数据永不过期的消息队列中查询流更改日志。

## 1.2 核心特性

**1）统一批处理和流处理**

批量写入和读取、流式更新、变更日志生成，全部支持。

**2）数据湖能力**

低成本、高可靠性、可扩展的元数据。 Apache Paimon 具有作为数据湖存储的所有优势。

**3）各种合并引擎**

按照您喜欢的方式更新记录。保留最后一条记录、进行部分更新或将记录聚合在一起，由您决定。

**4）变更日志生成**

Apache Paimon 可以从任何数据源生成正确且完整的变更日志，从而简化您的流分析。

**5）丰富的表类型**

除了主键表之外，Apache Paimon还支持append-only表，提供有序的流式读取来替代消息队列。

**6）模式演化**

Apache Paimon 支持完整的模式演化。您可以重命名列并重新排序。

## 1.3 基本概念

### 1.3.1 Snapshot

快照捕获表在某个时间点的状态。用户可以通过最新的快照来访问表的最新数据。通过时间旅行，用户还可以通过较早的快照访问表的先前状态。

### 1.3.2 Partition

Paimon 采用与 Apache Hive 相同的分区概念来分离数据。

分区是一种可选方法，可根据日期、城市和部门等特定列的值将表划分为相关部分。每个表可以有一个或多个分区键来标识特定分区。

通过分区，用户可以高效地操作表中的一片记录。

如果定义了主键，则分区键必须是主键的子集。

### 1.3.3 Bucket

未分区表或分区表中的分区被细分为存储桶，以便为可用于更有效查询的数据提供额外的结构。

桶的范围由记录中的一列或多列的哈希值确定。用户可以通过提供bucket-key选项来指定分桶列。如果未指定bucket-key选项，则主键（如果已定义）或完整记录将用作存储桶键。

桶是读写的最小存储单元，因此桶的数量限制了最大处理并行度。不过这个数字不应该太大，因为它会导致大量小文件和低读取性能。一般来说，建议每个桶的数据大小为1GB左右。

### 1.3.4 Consistency Guarantees一致性保证

Paimon writer使用两阶段提交协议以原子方式将一批记录提交到表中。每次提交在提交时最多生成两个快照。

对于任意两个同时修改表的writer，只要他们不修改同一个存储桶，他们的提交都是可序列化的。如果他们修改同一个存储桶，则仅保证快照隔离。也就是说，最终表状态可能是两次提交的混合，但不会丢失任何更改。

## 1.4 文件布局

一张表的所有文件都存储在一个基本目录下。 Paimon 文件以分层方式组织。下图说明了文件布局。从快照文件开始，Paimon 读者可以递归地访问表中的所有记录。

图示

描述已自动生成

下面简单介绍文件布局（不同操作对应文件如何变化，学习完Flink基本操作后再来理解，2.10进行分析）。

### 1.4.1 Snapshot Files

所有快照文件都存储在快照目录中。

快照文件是一个 JSON 文件，包含有关此快照的信息，包括：

正在使用的Schema文件

包含此快照的所有更改的清单列表（manifest list）

### 1.4.2 Manifest Files

所有清单列表（manifest list）和清单文件（manifest file）都存储在清单（manifest）目录中。

清单列表（manifest list）是清单文件名（manifest file）的列表。

清单文件（manifest file）是包含有关 LSM 数据文件和更改日志文件的文件信息。例如对应快照中创建了哪个LSM数据文件、删除了哪个文件。

### 1.4.3 Data Files

数据文件按分区和存储桶分组。每个存储桶目录都包含一个 LSM 树及其变更日志文件。目前，Paimon 支持使用 orc（默认）、parquet 和 avro 作为数据文件格式。

### 1.4.4 LSM Trees

Paimon 采用 LSM 树（日志结构合并树）作为文件存储的数据结构。

1.4.4.1 Sorted Runs

LSM 树将文件组织成多个Sorted Run。Sorted Run由一个或多个数据文件组成，并且每个数据文件恰好属于一个Sorted Run。

数据文件中的记录按其主键排序。在Sorted Run中，数据文件的主键范围永远不会重叠。

手机屏幕截图

描述已自动生成

正如您所看到的，不同的Sorted Run可能具有重叠的主键范围，甚至可能包含相同的主键。查询LSM树时，必须合并所有Sorted Run，并且必须根据用户指定的合并引擎和每条记录的时间戳来合并具有相同主键的所有记录。

写入LSM树的新记录将首先缓存在内存中。当内存缓冲区满时，内存中的所有记录将被排序并刷新到磁盘。

1.4.4.2 Compaction

当越来越多的记录写入LSM树时，Sorted Run的数量将会增加。由于查询LSM树需要将所有Sorted Run合并起来，太多Sorted Run将导致查询性能较差，甚至内存不足。

为了限制Sorted Run的数量，我们必须偶尔将多个Sorted Run合并为一个大的Sorted Run。这个过程称为Compaction。

然而，Compaction是一个资源密集型过程，会消耗一定的CPU时间和磁盘IO，因此过于频繁的Compaction可能会导致写入速度变慢。这是查询和写入性能之间的权衡。 Paimon 目前采用了类似于 Rocksdb 通用压缩的Compaction策略。

默认情况下，当Paimon将记录追加到LSM树时，它也会根据需要执行Compaction。用户还可以选择在“专用Compaction作业”中独立执行所有Compaction。

# 第2章 集成 Flink 引擎

Paimon目前支持Flink 1.17, 1.16, 1.15 和 1.14。本课程使用Flink 1.17.0。

## 2.1 环境准备

环境准备

### 2.1.1 安装 Flink

**1）上传并解压Flink安装包**

tar -zxvf flink-1.17.0-bin-scala\_2.12.tgz -C /opt/module/

**2）配置环境变量**

sudo vim /etc/profile.d/my\_env.sh

export HADOOP\_CLASSPATH=`hadoop classpath`

source /etc/profile.d/my\_env.sh

### 2.1.2 上传 jar 包

**1）下载并上传Paimon的jar包**

jar包下载地址：<https://repository.apache.org/snapshots/org/apache/paimon/paimon-flink-1.17/0.5-SNAPSHOT/>

**2）拷贝paimon的jar包到flink的lib目录下**

cp paimon-flink-1.17-0.5-20230703.002437-67.jar /opt/module/flink-1.17.0/lib

### 2.1.3 启动 Hadoop

（略）

### 2.1.4 启动 sql-client

**1）修改flink-conf.yaml配置**

vim /opt/module/flink-1.16.0/conf/flink-conf.yaml

#解决中文乱码，1.17之前参数是env.java.opts

env.java.opts.all: -Dfile.encoding=UTF-8

classloader.check-leaked-classloader: false

taskmanager.numberOfTaskSlots: 4

execution.checkpointing.interval: 10s

state.backend: rocksdb

state.checkpoints.dir: hdfs://hadoop102:8020/ckps

state.backend.incremental: true

**2）启动 Flink集群**

（1）解决依赖问题

cp /opt/module/hadoop-3.3.4/share/hadoop/mapreduce/hadoop-mapreduce-client-core-3.3.4.jar /opt/module/flink-1.17.0/lib/

（2）这里以 Yarn-Session模式为例

/opt/module/flink-1.17.0/bin/yarn-session.sh -d

**3）启动Flink的sql-client**

/opt/module/flink-1.17.0/bin/sql-client.sh -s yarn-session

图片包含 图示

描述已自动生成

**4）设置结果显示模式**

SET 'sql-client.execution.result-mode' = 'tableau';

## 2.2 Catalog

Paimon Catalog可以持久化元数据，当前支持两种类型的metastore：

* 文件系统（默认）：将元数据和表文件存储在文件系统中。
* hive：在 hive metastore中存储元数据。用户可以直接从 Hive 访问表。

### 2.2.1 文件系统

CREATE CATALOG fs\_catalog WITH (

'type' = 'paimon',

'warehouse' = 'hdfs://hadoop102:8020/paimon/fs'

);

USE CATALOG fs\_catalog;

### 2.2.2 Hive Catalog

通过使用Hive Catalog，对Catalog的更改将直接影响相应的hive metastore。在此类Catalog中创建的表也可以直接从 Hive 访问。

要使用 Hive Catalog，数据库名称、表名称和字段名称应小写。

**1）上传 hive-connector**

将flink-sql-connector-hive-3.1.3\_2.12-1.17.0.jar上川到Flink的lib目录下

**2）重启yarn-session集群**

**3）启动hive的metastore服务**

nohup hive --service metastore &

**4）创建Hive Catalog**

CREATE CATALOG hive\_catalog WITH (

'type' = 'paimon',

'metastore' = 'hive',

'uri' = 'thrift://hadoop102:9083',

'hive-conf-dir' = '/opt/module/hive/conf',

'warehouse' = 'hdfs://hadoop102:8020/paimon/hive'

);

USE CATALOG hive\_catalog;

**5）注意事项**

使用hive Catalog通过alter table更改不兼容的列类型时，参见 HIVE-17832。需要配置

vim /opt/module/hive/conf/hive-site.xml;

<property>

<name>hive.metastore.disallow.incompatible.col.type.changes</name>

<value>false</value>

</property>

上述配置需要在hive-site.xml中配置，且hive metastore服务需要重启。

如果使用的是 Hive3，请禁用 Hive ACID：

hive.strict.managed.tables=false

hive.create.as.insert.only=false

metastore.create.as.acid=false

### 2.2.3 sql 初始化文件

**1）创建初始化sql文件**

vim conf/sql-client-init.sql

CREATE CATALOG fs\_catalog WITH (

'type' = 'paimon',

'warehouse' = 'hdfs://hadoop102:8020/paimon/fs'

);

CREATE CATALOG hive\_catalog WITH (

'type' = 'paimon',

'metastore' = 'hive',

'uri' = 'thrift://hadoop102:9083',

'hive-conf-dir' = '/opt/module/hive/conf',

'warehouse' = 'hdfs://hadoop102:8020/paimon/hive'

);

USE CATALOG hive\_catalog;

SET 'sql-client.execution.result-mode' = 'tableau';

**2）启动sql-client时，指定该sql初始化文件**

bin/sql-client.sh -s yarn-session -i conf/sql-client-init.sql

**3）查看catalog**

show catalogs;

show current catalog;

## 2.3 DDL

### 2.3.1 建表

2.3.1.1 管理表

在 Paimon Catalog中创建的表就是Paimon的管理表，由Catalog管理。当表从Catalog中删除时，其表文件也将被删除，类似于Hive的内部表。

**1）创建表**

CREATE TABLE test (

user\_id BIGINT,

item\_id BIGINT,

behavior STRING,

dt STRING,

hh STRING,

PRIMARY KEY (dt, hh, user\_id) NOT ENFORCED

);

**2）创建分区表**

CREATE TABLE test\_p (

user\_id BIGINT,

item\_id BIGINT,

behavior STRING,

dt STRING,

hh STRING,

PRIMARY KEY (dt, hh, user\_id) NOT ENFORCED

) PARTITIONED BY (dt, hh);

通过配置partition.expiration-time，可以自动删除过期的分区。

如果定义了主键，则分区字段必须是主键的子集。

可以定义以下三类字段为分区字段：

* 创建时间（推荐）：创建时间通常是不可变的，因此您可以放心地将其视为分区字段并将其添加到主键中。
* 事件时间：事件时间是原表中的一个字段。对于CDC数据来说，比如从MySQL CDC同步的表或者Paimon生成的Changelogs，它们都是完整的CDC数据，包括UPDATE\_BEFORE记录，即使你声明了包含分区字段的主键，也能达到独特的效果。
* CDC op\_ts：不能定义为分区字段，无法知道之前的记录时间戳。

**3）Create Table As**

表可以通过查询的结果创建和填充，例如，我们有一个这样的sql： CREATE TABLE table\_b AS SELECT id, name FORM table\_a, 生成的表table\_b将相当于创建表并插入数据以下语句：CREATE TABLE table\_b(id INT, name STRING); INSERT INTO table\_b SELECT id, name FROM table\_a;

使用CREATE TABLE AS SELECT时我们可以指定主键或分区。

CREATE TABLE test1(

user\_id BIGINT,

item\_id BIGINT

);

CREATE TABLE test2 AS SELECT \* FROM test1;

-- 指定分区

CREATE TABLE test2\_p WITH ('partition' = 'dt') AS SELECT \* FROM test\_p;

-- 指定配置

CREATE TABLE test3(

user\_id BIGINT,

item\_id BIGINT

) WITH ('file.format' = 'orc');

CREATE TABLE test3\_op WITH ('file.format' = 'parquet') AS SELECT \* FROM test3;

-- 指定主键

CREATE TABLE test\_pk WITH ('primary-key' = 'dt,hh') AS SELECT \* FROM test;

-- 指定主键和分区

CREATE TABLE test\_all WITH ('primary-key' = 'dt,hh', 'partition' = 'dt') AS SELECT \* FROM test\_p;

**4）Create Table Like**

创建与另一个表具有相同schema、分区和表属性的表。

CREATE TABLE test\_ctl LIKE test;

**5）表属性**

用户可以指定表属性来启用Paimon的功能或提高Paimon的性能。有关此类属性的完整列表，请参阅配置: <https://paimon.apache.org/docs/master/maintenance/configurations/>。

CREATE TABLE tbl(

user\_id BIGINT,

item\_id BIGINT,

behavior STRING,

dt STRING,

hh STRING,

PRIMARY KEY (dt, hh, user\_id) NOT ENFORCED

) PARTITIONED BY (dt, hh)

WITH (

'bucket' = '2',

'bucket-key' = 'user\_id'

);

2.3.1.2 外部表

外部表由Catalog记录但不管理。如果删除外部表，其表文件不会被删除，类似于Hive的外部表。

Paimon 外部表可以在任何Catalog中使用。如果您不想创建Paimon Catalog而只想读/写表，则可以考虑外部表。

CREATE TABLE ex (

user\_id BIGINT,

item\_id BIGINT,

behavior STRING,

dt STRING,

hh STRING,

PRIMARY KEY (dt, hh, user\_id) NOT ENFORCED

) WITH (

'connector' = 'paimon',

'path' = 'hdfs://hadoop102:8020/paimon/external/ex',

'auto-create' = 'true'

);

2.3.1.3 临时表

仅 Flink 支持临时表。与外部表一样，临时表只是记录，但不由当前 Flink SQL 会话管理。如果临时表被删除，其资源将不会被删除。当 Flink SQL 会话关闭时，临时表也会被删除。与外部表的区别在于，临时表在Paimon Catalog中创建。

如果想将Paimon Catalog与其他表一起使用，但不想将它们存储在其他Catalog中，可以创建临时表。

USE CATALOG hive\_catalog;

CREATE TEMPORARY TABLE temp (

k INT,

v STRING

) WITH (

'connector' = 'filesystem',

'path' = 'hdfs://hadoop102:8020/temp.csv',

'format' = 'csv'

);

### 2.3.2 修改表

2.3.2.1 修改表

**1）更改/添加表属性**

ALTER TABLE test SET (

'write-buffer-size' = '256 MB'

);

**2）重命名表名称**

ALTER TABLE test1 RENAME TO test\_new;

**3）删除表属性**

ALTER TABLE test RESET ('write-buffer-size');

2.3.2.2 修改列

**1）添加新列**

ALTER TABLE test ADD (c1 INT, c2 STRING);

**2）重命名列名称**

ALTER TABLE test RENAME c1 TO c0;

**3）删除列**

ALTER TABLE test DROP (c0, c2);

**4）更改列的可为空性**

CREATE TABLE test\_null(

id INT PRIMARY KEY NOT ENFORCED,

coupon\_info FLOAT NOT NULL

);

-- 列coupon\_info修改成允许为null

ALTER TABLE test\_null MODIFY coupon\_info FLOAT;

-- 列coupon\_info修改成不允许为null

-- 如果表中已经有null值, 修改之前先设置如下参数删除null值

SET 'table.exec.sink.not-null-enforcer' = 'DROP';

ALTER TABLE test\_null MODIFY coupon\_info FLOAT NOT NULL;

**5）更改列注释**

ALTER TABLE test MODIFY user\_id BIGINT COMMENT 'user id';

**6）添加列位置**

ALTER TABLE test ADD a INT FIRST;

ALTER TABLE test ADD b INT AFTER a;

**7）更改列位置**

ALTER TABLE test MODIFY b INT FIRST;

ALTER TABLE test MODIFY a INT AFTER user\_id;

**8）更改列类型**

ALTER TABLE test MODIFY a DOUBLE;

2.3.2.3 修改水印

**1）添加水印**

CREATE TABLE test\_wm (

id INT,

name STRING,

ts BIGINT

);

ALTER TABLE test\_wm ADD(

et AS to\_timestamp\_ltz(ts,3),

WATERMARK FOR et AS et - INTERVAL '1' SECOND

);

**2）更改水印**

ALTER TABLE test\_wm MODIFY WATERMARK FOR et AS et - INTERVAL '2' SECOND;

**3）去掉水印**

ALTER TABLE test\_wm DROP WATERMARK;

## 2.4 DML

### 2.4.1 插入数据

INSERT 语句向表中插入新行或覆盖表中的现有数据。插入的行可以由值表达式或查询结果指定，跟标准的sql语法一致。

INSERT { INTO | OVERWRITE } table\_identifier [ part\_spec ] [ column\_list ] { value\_expr | query }

* part\_spec

可选，指定分区的键值对列表，多个用逗号分隔。可以使用类型文字（例如，date'2019-01-02'）。

语法： PARTITION (分区列名称 = 分区列值 [ , … ] )

* column\_list

可选，指定以逗号分隔的字段列表。

语法：(col\_name1 [,column\_name2, …])

所有指定的列都应该存在于表中，并且不能相互重复。它包括除静态分区列之外的所有列。字段列表的大小应与 VALUES 子句或查询中的数据大小完全相同。

* value\_expr

指定要插入的值。可以插入显式指定的值或 NULL。必须使用逗号分隔子句中的每个值。可以指定多于一组的值来插入多行。

语法：VALUES ( { 值 | NULL } [ , … ] ) [ , ( … ) ]

目前，Flink 不支持直接使用 NULL，因此需要将 NULL 转换为实际数据类型值，比如“CAST (NULL AS STRING)”

注意：将 Nullable 字段写入 Not-null 字段

不能将另一个表的可为空列插入到一个表的非空列中。Flink可以使用COALESCE函数来处理，比如A表的key1是not null，B表的key2是nullable：

INSERT INTO A key1 SELECT COALESCE(key2, <non-null expression>) FROM B

案例：

INSERT INTO test VALUES(1,1,'order','2023-07-01','1'), (2,2,'pay','2023-07-01','2');

INSERT INTO test\_p PARTITION(dt='2023-07-01',hh='1') VALUES(3,3, 'pv');

-- 执行模式区分流、批

INSERT INTO test\_p SELECT \* from test;

Paimon支持在sink阶段通过partition和bucket对数据进行shuffle。

### 2.4.2 覆盖数据

覆盖数据只支持batch模式。默认情况下，流式读取将忽略 INSERT OVERWRITE 生成的提交。如果你想读取OVERWRITE的提交，你可以配置streaming-read-overwrite。

RESET 'execution.checkpointing.interval';

SET 'execution.runtime-mode' = 'batch';

**1）覆盖未分区的表**

INSERT OVERWRITE test VALUES(3,3,'pay','2023-07-01','2');

**2）覆盖分区表**

对于分区表，Paimon默认的覆盖模式是动态分区覆盖（即Paimon只删除insert overwrite数据中出现的分区）。您可以配置动态分区覆盖来更改它。

INSERT OVERWRITE test\_p SELECT \* from test;

覆盖指定分区：

INSERT OVERWRITE test\_p PARTITION (dt = '2023-07-01', hh = '2') SELECT user\_id,item\_id,behavior from test;

**3）清空表**

可以使用 INSERT OVERWRITE 通过插入空值来清除表（关闭动态分区覆盖）。

INSERT OVERWRITE test\_p/\*+ OPTIONS('dynamic-partition-overwrite'='false') \*/ SELECT \* FROM test\_p WHERE false;

### 2.4.3 更新数据

目前，Paimon 在 Flink 1.17 及后续版本中支持使用 UPDATE 更新记录。您可以在Flink的批处理模式下执行UPDATE。

只有主键表支持此功能。不支持更新主键。

MergeEngine 需要deduplicate或partial-update才能支持此功能。（默认deduplicate）

UPDATE test SET item\_id = 4, behavior = 'pv' WHERE user\_id = 3;

### 2.4.4 删除数据

从表中删除（Flink 1.17）：

* 只有写入模式设置为change-log的表支持此功能。（有主键默认就是change-log）
* 如果表有主键，MergeEngine需要为deduplicate。（默认deduplicate）

DELETE FROM test WHERE user\_id = 3;

### 2.4.5 Merge Into

通过merge into实现行级更新，只有主键表支持此功能。该操作不会产生 UPDATE\_BEFORE，因此不建议设置 'changelog-producer' = 'input'。

merge-into 操作使用“upsert”语义而不是“update”，这意味着如果该行存在，则执行更新，否则执行插入。

**1）语法说明：**

<FLINK\_HOME>/bin/flink run \

/path/to/paimon-flink-action-0.5-SNAPSHOT.jar \

merge-into \

--warehouse <warehouse-path> \

--database <database-name> \

--table <target-table> \

[--target-as <target-table-alias>] \

--source-table <source-table-name> \

[--source-sql <sql> ...]\

--on <merge-condition> \

--merge-actions <matched-upsert,matched-delete,not-matched-insert,not-matched-by-source-upsert,not-matched-by-source-delete> \

--matched-upsert-condition <matched-condition> \

--matched-upsert-set <upsert-changes> \

--matched-delete-condition <matched-condition> \

--not-matched-insert-condition <not-matched-condition> \

--not-matched-insert-values <insert-values> \

--not-matched-by-source-upsert-condition <not-matched-by-source-condition> \

--not-matched-by-source-upsert-set <not-matched-upsert-changes> \

--not-matched-by-source-delete-condition <not-matched-by-source-condition> \

[--catalog-conf <paimon-catalog-conf> [--catalog-conf <paimon-catalog-conf> ...]]

--source-sql <sql> 可以传递sql来配置环境并在运行时创建源表。

“match”的说明：

（1）matched：更改的行来自目标表，每个行都可以根据条件匹配源表行（source ∩ target）：

* 合并条件（--on）
* 匹配条件（--matched-xxx-condition）

（2）not-matched：更改的行来自源表，并且根据条件所有行都不能与任何目标表的行匹配（source – target）：

* 合并条件（--on）
* 不匹配条件（--not-matched-xxx-condition）：不能使用目标表的列来构造条件表达式。

（3）not-matched-by-source：更改的行来自目标表，并且基于条件所有行都不能与任何源表的行匹配（target – source）：

* 合并条件（--on）
* 源不匹配条件（--not-matched-by-source-xxx-condition）：不能使用源表的列来构造条件表达式。

**2）案例实操**

需要用到paimon-flink-action-xxxx.jar，上传：

cp paimon-flink-action-0.5-20230703.002437-53.jar /opt/module/flink-1.17.0/opt

下载地址：

<https://repository.apache.org/snapshots/org/apache/paimon/paimon-flink-action/0.5-SNAPSHOT/>

（1）准备测试表：

use catalog hive\_catalog;

create database test;

use test;

CREATE TABLE ws1 (

id INT,

ts BIGINT,

vc INT,

PRIMARY KEY (id) NOT ENFORCED

);

INSERT INTO ws1 VALUES(1,1,1),(2,2,2),(3,3,3);

CREATE TABLE ws\_t (

id INT,

ts BIGINT,

vc INT,

PRIMARY KEY (id) NOT ENFORCED

);

INSERT INTO ws\_t VALUES(2,2,2),(3,3,3),(4,4,4),(5,5,5);

（2）案例一： ws\_t与ws1匹配id，将ws\_t中ts>2的vc改为10，ts<=2的删除

bin/flink run \

/opt/module/flink-1.17.0/opt/paimon-flink-action-0.5-20230703.002437-53.jar \

merge-into \

--warehouse hdfs://hadoop102:8020/paimon/hive \

--database test \

--table ws\_t \

--source-table test.ws1 \

--on "ws\_t.id = ws1.id" \

--merge-actions matched-upsert,matched-delete \

--matched-upsert-condition "ws\_t.ts > 2" \

--matched-upsert-set "vc = 10" \

--matched-delete-condition "ws\_t.ts <= 2"

（3）案例二： ws\_t与ws1匹配id，匹配上的将ws\_t中vc加10，ws1中没匹配上的插入ws\_t中

bin/flink run \

/opt/module/flink-1.17.0/opt/paimon-flink-action-0.5-20230703.002437-53.jar \

merge-into \

--warehouse hdfs://hadoop102:8020/paimon/hive \

--database test \

--table ws\_t \

--source-table test.ws1 \

--on "ws\_t.id = ws1.id" \

--merge-actions matched-upsert,not-matched-insert \

--matched-upsert-set "vc = ws\_t.vc + 10" \

--not-matched-insert-values "\*"

（4）案例三： ws\_t与ws1匹配id，ws\_t中没匹配上的，ts大于4则vc加20，ts=4则删除

bin/flink run \

/opt/module/flink-1.17.0/opt/paimon-flink-action-0.5-20230703.002437-53.jar \

merge-into \

--warehouse hdfs://hadoop102:8020/paimon/hive \

--database test \

--table ws\_t \

--source-table test.ws1 \

--on "ws\_t.id = ws1.id" \

--merge-actions not-matched-by-source-upsert,not-matched-by-source-delete \

--not-matched-by-source-upsert-condition "ws\_t.ts > 4" \

--not-matched-by-source-upsert-set "vc = ws\_t.vc + 20" \

--not-matched-by-source-delete-condition " ws\_t.ts = 4"

（5）案例四： 使用--source-sql创建新catalog下的源表，匹配ws\_t的id，没匹配上的插入ws\_t

bin/flink run \

/opt/module/flink-1.17.0/opt/paimon-flink-action-0.5-20230703.002437-53.jar \

merge-into \

--warehouse hdfs://hadoop102:8020/paimon/hive \

--database test \

--table ws\_t \

--source-sql "CREATE CATALOG fs2 WITH ('type' = 'paimon','warehouse' = 'hdfs://hadoop102:8020/paimon/fs2')" \

--source-sql "CREATE DATABASE IF NOT EXISTS fs2.test" \

--source-sql "CREATE TEMPORARY VIEW fs2.test.ws2 AS SELECT id+10 as id,ts,vc FROM test.ws1" \

--source-table fs2.test.ws2 \

--on "ws\_t.id = ws2. id" \

--merge-actions not-matched-insert\

--not-matched-insert-values "\*"

## 2.5 DQL查询表

### 2.5.1 批量查询

就像所有其他表一样，Paimon 表可以使用 SELECT 语句进行查询。

Paimon的批量读取返回表快照中的所有数据。默认情况下，批量读取返回最新快照。

在sql-client中，设置执行模式为批即可：

RESET 'execution.checkpointing.interval';

SET 'execution.runtime-mode' = 'batch';

2.5.1.1 时间旅行

**1）读取指定id的快照**

SELECT \* FROM ws\_t /\*+ OPTIONS('scan.snapshot-id' = '1') \*/;

SELECT \* FROM ws\_t /\*+ OPTIONS('scan.snapshot-id' = '2') \*/;

**2）读取指定时间戳的快照**

-- 查看快照信息

SELECT \* FROM ws\_t&snapshots;

SELECT \* FROM ws\_t /\*+ OPTIONS('scan.timestamp-millis' = '1688369660841') \*/;

**3）读取指定标签**

SELECT \* FROM ws\_t /\*+ OPTIONS('scan.tag-name' = 'my-tag') \*/;

2.5.1.2 增量查询

读取开始快照（不包括）和结束快照之间的增量更改。例如，“3,5”表示快照 3 和快照 5 之间的更改：

SELECT \* FROM ws\_t /\*+ OPTIONS('incremental-between' = '3,5') \*/;

在batch模式中，不返回DELETE记录，因此-D的记录将被删除。如果你想查看DELETE记录，可以查询audit\_log表：

SELECT \* FROM ws\_t$audit\_log /\*+ OPTIONS('incremental-between' = '3,5') \*/;

### 2.5.2 流式查询

默认情况下，Streaming read 在第一次启动时会生成表上的最新快照，并继续读取最新的更改。

SET 'execution.checkpointing.interval'='30s';

SET 'execution.runtime-mode' = 'streaming';

也可以从最新读取，设置扫描模式：

SELECT \* FROM ws\_t /\*+ OPTIONS('scan.mode' = 'latest') \*/

2.5.2.1 时间旅行

如果只想处理今天及以后的数据，则可以使用分区过滤器来实现：

SELECT \* FROM test\_p WHERE dt > '2023-07-01'

如果不是分区表，或者无法按分区筛选，可以使用时间旅行的流读取。

**1）从指定快照id开始读取变更数据**

SELECT \* FROM ws\_t /\*+ OPTIONS('scan.snapshot-id' = '1') \*/;

**2）从指定时间戳开始读取**

SELECT \* FROM ws\_t /\*+ OPTIONS('scan.timestamp-millis' = '1688369660841') \*/;

**3）第一次启动时读取指定快照数据，并继续读取变化**

SELECT \* FROM ws\_t /\*+ OPTIONS('scan.mode'='from-snapshot-full','scan.snapshot-id' = '3') \*/;

2.5.2.2 Consumer ID

**1）优点**

在流式读取表时指定consumer-id，这是一个实验性功能。

当流读取Paimon表时，下一个快照id将被记录到文件系统中。这有几个优点：

* 当之前的作业停止后，新启动的作业可以继续消耗之前的进度，而不需要从状态恢复。新的读取将从消费者文件中找到的下一个快照 ID 开始读取。
* 在判断一个快照是否过期时，Paimon会查看文件系统中该表的所有消费者，如果还有消费者依赖这个快照，那么这个快照就不会因为过期而被删除。
* 当没有水印定义时，Paimon表会将快照中的水印传递到下游Paimon表，这意味着您可以跟踪整个管道的水印进度。

注意：消费者将防止快照过期。可以指定“consumer.expiration-time”来管理消费者的生命周期。

**2）案例演示**

指定consumer-id开始流式查询：

SELECT \* FROM ws\_t /\*+ OPTIONS('consumer-id' = 'atguigu') \*/;

停掉原先的流式查询，插入数据：

insert into ws\_t values(6,6,6);

再次指定consumer-id流式查询：

SELECT \* FROM ws\_t /\*+ OPTIONS('consumer-id' = 'atguigu') \*/;

### 2.5.3 查询优化

强烈建议在查询时指定分区和主键过滤器，这将加快查询的数据跳过速度。

可以加速数据跳跃的过滤函数有：

* =
* <
* <=
* >
* >=
* IN (...)
* LIKE 'abc%'
* IS NULL

Paimon会按主键对数据进行排序，从而加快点查询和范围查询的速度。使用复合主键时，查询过滤器最好形成主键的最左边前缀，以获得良好的加速效果。

CREATE TABLE orders (

catalog\_id BIGINT,

order\_id BIGINT,

.....,

PRIMARY KEY (catalog\_id, order\_id) NOT ENFORCED -- composite primary key

)

通过为主键最左边的前缀指定范围过滤器，查询获得了很好的加速。

SELECT \* FROM orders WHERE catalog\_id=1025;

SELECT \* FROM orders WHERE catalog\_id=1025 AND order\_id=29495;

SELECT \* FROM orders

WHERE catalog\_id=1025jkjkjk

AND order\_id>2035 AND order\_id<6000;

下面例子的过滤器不能很好地加速查询：

SELECT \* FROM orders WHERE order\_id=29495;

SELECT \* FROM orders WHERE catalog\_id=1025 OR order\_id=29495;

## 2.6 系统表

系统表包含有关每个表的元数据和信息，例如创建的快照和使用的选项。用户可以通过批量查询访问系统表。

### 2.6.1 快照表 Snapshots Table

通过snapshots表可以查询表的快照历史信息，包括快照中发生的记录数。

SELECT \* FROM ws\_t$snapshots;

通过查询快照表，可以了解该表的提交和过期信息以及数据的时间旅行。

### 2.6.2 模式表 Schemas Table

通过schemas表可以查询该表的历史schema。

SELECT \* FROM ws\_t$schemas;

可以连接快照表和模式表以获取给定快照的字段。

SELECT s.snapshot\_id, t.schema\_id, t.fields

FROM ws\_t$snapshots s JOIN ws\_t$schemas t

ON s.schema\_id=t.schema\_id where s.snapshot\_id=3;

### 2.6.3 选项表 Options Table

可以通过选项表查询DDL中指定的表的选项信息。未显示的选项将是默认值。

SELECT \* FROM ws\_t$options;

### 2.6.4 审计日志表 Audit log Table

如果需要审计表的changelog，可以使用audit\_log系统表。通过audit\_log表，获取表增量数据时可以获取rowkind列。您可以利用该栏目进行过滤等操作来完成审核。

rowkind 有四个值：

+I：插入操作。

-U：使用更新行的先前内容进行更新操作。

+U：使用更新行的新内容进行更新操作。

-D：删除操作。

SELECT \* FROM ws\_t$audit\_log;

### 2.6.5 文件表 Files Table

可以查询特定快照表的文件。

-- 查询最新快照的文件

SELECT \* FROM ws\_t$files;

-- 查询指定快照的文件

SELECT \* FROM ws\_t$files /\*+ OPTIONS('scan.snapshot-id'='1') \*/;

### 2.6.6 标签表 Tags Table

通过tags表可以查询表的标签历史信息，包括基于哪些快照进行标签以及快照的一些历史信息。您还可以通过名称获取所有标签名称和时间旅行到特定标签的数据。

SELECT \* FROM ws\_t$tags;

## 2.7 维表Join

Paimon支持Lookup Join语法，它用于从 Paimon 查询的数据来补充维度字段。要求一个表具有处理时间属性，而另一个表由查找源连接器支持。

Paimon 支持 Flink 中具有主键的表和append-only的表查找联接。以下示例说明了此功能。

USE CATALOG fs\_catalog;

CREATE TABLE customers (

id INT PRIMARY KEY NOT ENFORCED,

name STRING,

country STRING,

zip STRING

);

INSERT INTO customers VALUES(1,'zs','ch','123'),(2,'ls','ch','456'), (3,'ww','ch','789');

CREATE TEMPORARY TABLE Orders (

order\_id INT,

total INT,

customer\_id INT,

proc\_time AS PROCTIME()

) WITH (

'connector' = 'datagen',

'rows-per-second'='1',

'fields.order\_id.kind'='sequence',

'fields.order\_id.start'='1',

'fields.order\_id.end'='1000000',

'fields.total.kind'='random',

'fields.total.min'='1',

'fields.total.max'='1000',

'fields.customer\_id.kind'='random',

'fields.customer\_id.min'='1',

'fields.customer\_id.max'='3'

);

SELECT o.order\_id, o.total, c.country, c.zip

FROM Orders AS o

JOIN customers

FOR SYSTEM\_TIME AS OF o.proc\_time AS c

ON o.customer\_id = c.id;

Lookup Join算子会在本地维护一个RocksDB缓存并实时拉取表的最新更新。查找连接运算符只会提取必要的数据，因此您的过滤条件对于性能非常重要。

如果Orders（主表）的记录Join缺失，因为customers（查找表）对应的数据还没有准备好。可以考虑使用Flink的Delayed Retry Strategy For Lookup。

## 2.8 CDC集成

Paimon 支持多种通过模式演化将数据提取到 Paimon 表中的方法。这意味着添加的列会实时同步到Paimon表中，并且不会为此重新启动同步作业。

目前支持以下同步方式：

* MySQL同步表：将MySQL中的一张或多张表同步到一张Paimon表中。
* MySQL同步数据库：将整个MySQL数据库同步到一个Paimon数据库中。
* API同步表：将您的自定义DataStream输入同步到一张Paimon表中。
* Kafka同步表：将一个Kafka topic的表同步到一张Paimon表中。
* Kafka同步数据库：将一个包含多表的Kafka主题或多个各包含一表的主题同步到一个Paimon数据库中。

### 2.8.1 MySQL

添加Flink CDC 连接器。

cp flink-sql-connector-mysql-cdc-2.4.0.jar /opt/module/flink-1.17.0/lib

重启yarn-session集群和sql-client。

2.8.1.1 同步表

**1）语法说明**

<FLINK\_HOME>/bin/flink run \

/path/to/paimon-flink-action-0.5-SNAPSHOT.jar \

mysql-sync-table

--warehouse <warehouse-path> \

--database <database-name> \

--table <table-name> \

[--partition-keys <partition-keys>] \

[--primary-keys <primary-keys>] \

[--computed-column <'column-name=expr-name(args[, ...])'> [--computed-column ...]] \

[--mysql-conf <mysql-cdc-source-conf> [--mysql-conf <mysql-cdc-source-conf> ...]] \

[--catalog-conf <paimon-catalog-conf> [--catalog-conf <paimon-catalog-conf> ...]] \

[--table-conf <paimon-table-sink-conf> [--table-conf <paimon-table-sink-conf> ...]]

参数说明：

| **配置** | **描述** |
| --- | --- |
| --warehouse | Paimon仓库路径。 |
| --database | Paimon Catalog中的数据库名称。 |
| --table | Paimon 表名称。 |
| --partition-keys | Paimon 表的分区键。如果有多个分区键，请用逗号连接，例如“dt,hh,mm”。 |
| --primary-keys | Paimon 表的主键。如果有多个主键，请用逗号连接，例如“buyer\_id,seller\_id”。 |
| --computed-column | 计算列的定义。参数字段来自 MySQL 表字段名称。 |
| --mysql-conf | Flink CDC MySQL 源表的配置。每个配置都应以“key=value”的格式指定。主机名、用户名、密码、数据库名和表名是必需配置，其他是可选配置。 |
| --catalog-conf | Paimon Catalog的配置。每个配置都应以“key=value”的格式指定。 |
| --table-conf | Paimon 表sink的配置。每个配置都应以“key=value”的格式指定。 |

如果指定的 Paimon 表不存在，此操作将自动创建该表。其schema将从所有指定的 MySQL 表派生。如果 Paimon 表已存在，则其schema将与所有指定 MySQL 表的schema进行比较。

**2）案例实操**

（1）MySQL一张表同步到Paimon一张表

bin/flink run \

/opt/module/flink-1.17.0/opt/paimon-flink-action-0.5-20230703.002437-53.jar \

mysql-sync-table \

--warehouse hdfs://hadoop102:8020/paimon/hive \

--database test \

--table order\_info\_cdc \

--primary-keys id \

--mysql-conf hostname=hadoop102 \

--mysql-conf username=root \

--mysql-conf password=000000 \

--mysql-conf database-name=gmall \

--mysql-conf table-name='order\_info' \

--catalog-conf metastore=hive \

--catalog-conf uri=thrift://hadoop102:9083 \

--table-conf bucket=4 \

--table-conf changelog-producer=input \

--table-conf sink.parallelism=4

（2）MySQL多张表同步到Paimon一张表

bin/flink run \

/opt/module/flink-1.17.0/opt/paimon-flink-action-0.5-20230703.002437-53.jar \

mysql-sync-table \

--warehouse hdfs://hadoop102:8020/paimon/hive \

--database test \

--table order\_cdc \

--primary-keys id \

--mysql-conf hostname=hadoop102 \

--mysql-conf username=root \

--mysql-conf password=000000 \

--mysql-conf database-name=gmall \

--mysql-conf table-name='order\_.\*' \

--catalog-conf metastore=hive \

--catalog-conf uri=thrift://hadoop102:9083 \

--table-conf bucket=4 \

--table-conf changelog-producer=input \

--table-conf sink.parallelism=4

2.8.1.2 同步数据库

**1）语法说明**

<FLINK\_HOME>/bin/flink run \

/path/to/paimon-flink-action-0.5-SNAPSHOT.jar \

mysql-sync-database

--warehouse <warehouse-path> \

--database <database-name> \

[--ignore-incompatible <true/false>] \

[--table-prefix <paimon-table-prefix>] \

[--table-suffix <paimon-table-suffix>] \

[--including-tables <mysql-table-name|name-regular-expr>] \

[--excluding-tables <mysql-table-name|name-regular-expr>] \

[--mysql-conf <mysql-cdc-source-conf> [--mysql-conf <mysql-cdc-source-conf> ...]] \

[--catalog-conf <paimon-catalog-conf> [--catalog-conf <paimon-catalog-conf> ...]] \

[--table-conf <paimon-table-sink-conf> [--table-conf <paimon-table-sink-conf> ...]]

参数说明：

| **配置** | **描述** |
| --- | --- |
| --warehouse | Paimon仓库路径。 |
| --database | Paimon Catalog中的数据库名称。 |
| --ignore-incompatible | 默认为 false，在这种情况下，如果 Paimon 中存在 MySQL 表名，并且它们的 schema 不兼容，则会抛出异常。您可以显式将其指定为 true 以忽略不兼容的表和异常。 |
| --table-prefix | 所有需要同步的Paimon表的前缀。例如，如果您希望所有同步表都以“ods\_”作为前缀，则可以指定“--table-prefix ods\_”。 |
| --table-suffix | 所有需要同步的Paimon表的后缀。用法与“--table-prefix”相同。 |
| --including-tables | 用于指定要同步哪些源表。您必须使用“|”分隔多个表，例如：'a|b|c'。支持正则表达式，例如指定“--include-tables test|paimon.\*”表示同步表'test'和所有表都以“paimon”开头。 |
| --excluding-tables | 用于指定哪些源表不同步。用法与“--include-tables”相同。如果同时指定了“-- except-tables”，则“-- except-tables”的优先级高于“--include-tables”。 |
| --mysql-conf | Flink CDC MySQL源表的配置。每个配置都应以“key=value”的格式指定。主机名、用户名、密码、数据库名和表名是必需配置，其他是可选配置。 |
| --catalog-conf | Paimon Catalog的配置。每个配置都应以“key=value”的格式指定。 |
| --table-conf | Paimon 表sink的配置。每个配置都应以“key=value”的格式指定。 |

只有具有主键的表才会被同步。

对于每个需要同步的MySQL表，如果对应的Paimon表不存在，该操作会自动创建该表。其schema将从所有指定的 MySQL 表派生。如果 Paimon 表已存在，则其schema将与所有指定 MySQL 表的schema进行比较。

**2）案例实操**

bin/flink run \

/opt/module/flink-1.17.0/opt/paimon-flink-action-0.5-20230703.002437-53.jar \

mysql-sync-database \

--warehouse hdfs://hadoop102:8020/paimon/hive \

--database test \

--table-prefix "ods\_" \

--table-suffix "\_cdc" \

--mysql-conf hostname=hadoop102 \

--mysql-conf username=root \

--mysql-conf password=000000 \

--mysql-conf database-name=gmall \

--catalog-conf metastore=hive \

--catalog-conf uri=thrift://hadoop102:9083 \

--table-conf bucket=4 \

--table-conf changelog-producer=input \

--table-conf sink.parallelism=4 \

--including-tables 'user\_info|order\_info|activity\_rule'

**3）同步数据库下新添加的表**

首先假设 Flink 作业正在同步数据库 source\_db 下的表 [product、user、address]。提交作业的命令如下所示：

<FLINK\_HOME>/bin/flink run \

/path/to/paimon-flink-action-0.5-SNAPSHOT.jar \

mysql-sync-database \

--warehouse hdfs:///path/to/warehouse \

--database test\_db \

--mysql-conf hostname=127.0.0.1 \

--mysql-conf username=root \

--mysql-conf password=123456 \

--mysql-conf database-name=source\_db \

--catalog-conf metastore=hive \

--catalog-conf uri=thrift://hive-metastore:9083 \

--table-conf bucket=4 \

--table-conf changelog-producer=input \

--table-conf sink.parallelism=4 \

--including-tables 'product|user|address'

稍后，我们希望作业也同步包含历史数据的表 [order, custom]。我们可以通过从作业的先前快照中恢复并从而重用作业的现有状态来实现这一点。恢复的作业将首先对新添加的表进行快照，然后自动从之前的位置继续读取变更日志。

从以前的快照恢复并添加新表进行同步的命令如下所示：

<FLINK\_HOME>/bin/flink run \

--fromSavepoint savepointPath \

/path/to/paimon-flink-action-0.5-SNAPSHOT.jar \

mysql-sync-database \

--warehouse hdfs:///path/to/warehouse \

--database test\_db \

--mysql-conf hostname=127.0.0.1 \

--mysql-conf username=root \

--mysql-conf password=123456 \

--mysql-conf database-name=source\_db \

--catalog-conf metastore=hive \

--catalog-conf uri=thrift://hive-metastore:9083 \

--table-conf bucket=4 \

--including-tables 'product|user|address|order|custom'

### 2.8.2 Kafka

Flink 提供了几种 Kafka CDC 格式：canal-json、debezium-json、ogg-json、maxwell-json。如果 Kafka 主题中的消息是使用更改数据捕获 (CDC) 工具从另一个数据库捕获的更改事件，则您可以使用 Paimon Kafka CDC。将解析后的INSERT、UPDATE、DELETE消息写入到paimon表中。Paimon官网列出支持的格式如下：

表格

描述已自动生成

添加Kafka连接器：

cp flink-sql-connector-kafka-1.17.0.jar /opt/module/flink-1.17.0/lib

重启yarn-session集群和sql-client。

2.8.2.1 同步表

**1）语法说明**

将 Kafka 的一个主题中的一张或多张表同步到一张 Paimon 表中。

<FLINK\_HOME>/bin/flink run \

/path/to/paimon-flink-action-0.5-SNAPSHOT.jar \

kafka-sync-table

--warehouse <warehouse-path> \

--database <database-name> \

--table <table-name> \

[--partition-keys <partition-keys>] \

[--primary-keys <primary-keys>] \

[--computed-column <'column-name=expr-name(args[, ...])'> [--computed-column ...]] \

[--kafka-conf <kafka-source-conf> [--kafka-conf <kafka-source-conf> ...]] \

[--catalog-conf <paimon-catalog-conf> [--catalog-conf <paimon-catalog-conf> ...]] \

[--table-conf <paimon-table-sink-conf> [--table-conf <paimon-table-sink-conf> ...]]

参数说明

| **配置** | **描述** |
| --- | --- |
| --warehouse | Paimon仓库路径。 |
| --database | Paimon Catalog中的数据库名称。 |
| --table | Paimon 表名称。 |
| --partition-keys | Paimon 表的分区键。如果有多个分区键，请用逗号连接，例如“dt,hh,mm”。 |
| --primary-keys | Paimon 表的主键。如果有多个主键，请用逗号连接，例如“buyer\_id,seller\_id”。 |
| --computed-column | 计算列的定义。参数字段来自 Kafka 主题的表字段名称。 |
| --kafka-conf | Flink Kafka 源的配置。每个配置都应以“key=value”的格式指定。 `properties.bootstrap.servers`、`topic`、`properties.group.id` 和 `value.format` 是必需配置，其他配置是可选的。 |
| --catalog-conf | Paimon Catalog的配置。每个配置都应以“key=value”的格式指定。 |
| --table-conf | Paimon 表sink的配置。每个配置都应以“key=value”的格式指定。 |

如果您指定的 Paimon 表不存在，此操作将自动创建该表。它的schema将从所有指定的Kafka topic的表中派生出来，它从topic中获取最早的非DDL数据解析schema。如果 Paimon 表已存在，则其schema将与所有指定 Kafka 主题表的schema进行比较。

**2）案例实操**

（1）准备数据（canal-json格式）

为了方便，直接将canal格式的数据插入topic里（user\_info单表数据）：

kafka-console-producer.sh --broker-list hadoop102:9092 --topic paimon\_canal

#插入数据如下：

{"data":[{"id":"6","login\_name":"t7dk2h","nick\_name":"冰冰11","passwd":null,"name":"淳于冰","phone\_num":"13178654378","email":"t7dk2h@263.net","head\_img":null,"user\_level":"1","birthday":"1997-12-08","gender":null,"create\_time":"2022-06-08 00:00:00","operate\_time":null,"status":null}],"database":"gmall","es":1689150607000,"id":1,"isDdl":false,"mysqlType":{"id":"bigint","login\_name":"varchar(200)","nick\_name":"varchar(200)","passwd":"varchar(200)","name":"varchar(200)","phone\_num":"varchar(200)","email":"varchar(200)","head\_img":"varchar(200)","user\_level":"varchar(200)","birthday":"date","gender":"varchar(1)","create\_time":"datetime","operate\_time":"datetime","status":"varchar(200)"},"old":[{"nick\_name":"冰冰"}],"pkNames":["id"],"sql":"","sqlType":{"id":-5,"login\_name":12,"nick\_name":12,"passwd":12,"name":12,"phone\_num":12,"email":12,"head\_img":12,"user\_level":12,"birthday":91,"gender":12,"create\_time":93,"operate\_time":93,"status":12},"table":"user\_info","ts":1689151566836,"type":"UPDATE"}

{"data":[{"id":"7","login\_name":"vihcj30p1","nick\_name":"豪心22","passwd":null,"name":"魏豪心","phone\_num":"13956932645","email":"vihcj30p1@live.com","head\_img":null,"user\_level":"1","birthday":"1991-06-07","gender":"M","create\_time":"2022-06-08 00:00:00","operate\_time":null,"status":null}],"database":"gmall","es":1689151623000,"id":2,"isDdl":false,"mysqlType":{"id":"bigint","login\_name":"varchar(200)","nick\_name":"varchar(200)","passwd":"varchar(200)","name":"varchar(200)","phone\_num":"varchar(200)","email":"varchar(200)","head\_img":"varchar(200)","user\_level":"varchar(200)","birthday":"date","gender":"varchar(1)","create\_time":"datetime","operate\_time":"datetime","status":"varchar(200)"},"old":[{"nick\_name":"豪心"}],"pkNames":["id"],"sql":"","sqlType":{"id":-5,"login\_name":12,"nick\_name":12,"passwd":12,"name":12,"phone\_num":12,"email":12,"head\_img":12,"user\_level":12,"birthday":91,"gender":12,"create\_time":93,"operate\_time":93,"status":12},"table":"user\_info","ts":1689151623139,"type":"UPDATE"}

{"data":[{"id":"8","login\_name":"02r2ahx","nick\_name":"卿卿33","passwd":null,"name":"穆卿","phone\_num":"13412413361","email":"02r2ahx@sohu.com","head\_img":null,"user\_level":"1","birthday":"2001-07-08","gender":"F","create\_time":"2022-06-08 00:00:00","operate\_time":null,"status":null}],"database":"gmall","es":1689151626000,"id":3,"isDdl":false,"mysqlType":{"id":"bigint","login\_name":"varchar(200)","nick\_name":"varchar(200)","passwd":"varchar(200)","name":"varchar(200)","phone\_num":"varchar(200)","email":"varchar(200)","head\_img":"varchar(200)","user\_level":"varchar(200)","birthday":"date","gender":"varchar(1)","create\_time":"datetime","operate\_time":"datetime","status":"varchar(200)"},"old":[{"nick\_name":"卿卿"}],"pkNames":["id"],"sql":"","sqlType":{"id":-5,"login\_name":12,"nick\_name":12,"passwd":12,"name":12,"phone\_num":12,"email":12,"head\_img":12,"user\_level":12,"birthday":91,"gender":12,"create\_time":93,"operate\_time":93,"status":12},"table":"user\_info","ts":1689151626863,"type":"UPDATE"}

{"data":[{"id":"9","login\_name":"mjhrxnu","nick\_name":"武新44","passwd":null,"name":"罗武新","phone\_num":"13617856358","email":"mjhrxnu@yahoo.com","head\_img":null,"user\_level":"1","birthday":"2001-08-08","gender":null,"create\_time":"2022-06-08 00:00:00","operate\_time":null,"status":null}],"database":"gmall","es":1689151630000,"id":4,"isDdl":false,"mysqlType":{"id":"bigint","login\_name":"varchar(200)","nick\_name":"varchar(200)","passwd":"varchar(200)","name":"varchar(200)","phone\_num":"varchar(200)","email":"varchar(200)","head\_img":"varchar(200)","user\_level":"varchar(200)","birthday":"date","gender":"varchar(1)","create\_time":"datetime","operate\_time":"datetime","status":"varchar(200)"},"old":[{"nick\_name":"武新"}],"pkNames":["id"],"sql":"","sqlType":{"id":-5,"login\_name":12,"nick\_name":12,"passwd":12,"name":12,"phone\_num":12,"email":12,"head\_img":12,"user\_level":12,"birthday":91,"gender":12,"create\_time":93,"operate\_time":93,"status":12},"table":"user\_info","ts":1689151630781,"type":"UPDATE"}

{"data":[{"id":"10","login\_name":"kwua2155","nick\_name":"纨纨55","passwd":null,"name":"姜纨","phone\_num":"13742843828","email":"kwua2155@163.net","head\_img":null,"user\_level":"3","birthday":"1997-11-08","gender":"F","create\_time":"2022-06-08 00:00:00","operate\_time":null,"status":null}],"database":"gmall","es":1689151633000,"id":5,"isDdl":false,"mysqlType":{"id":"bigint","login\_name":"varchar(200)","nick\_name":"varchar(200)","passwd":"varchar(200)","name":"varchar(200)","phone\_num":"varchar(200)","email":"varchar(200)","head\_img":"varchar(200)","user\_level":"varchar(200)","birthday":"date","gender":"varchar(1)","create\_time":"datetime","operate\_time":"datetime","status":"varchar(200)"},"old":[{"nick\_name":"纨纨"}],"pkNames":["id"],"sql":"","sqlType":{"id":-5,"login\_name":12,"nick\_name":12,"passwd":12,"name":12,"phone\_num":12,"email":12,"head\_img":12,"user\_level":12,"birthday":91,"gender":12,"create\_time":93,"operate\_time":93,"status":12},"table":"user\_info","ts":1689151633697,"type":"UPDATE"}

（2）从一个 Kafka 主题（包含单表数据）同步到 Paimon表

bin/flink run \

/opt/module/flink-1.17.0/opt/paimon-flink-action-0.5-20230703.002437-53.jar \

kafka-sync-table \

--warehouse hdfs://hadoop102:8020/paimon/hive \

--database test \

--table kafka\_user\_info\_cdc \

--primary-keys id \

--kafka-conf properties.bootstrap.servers=hadoop102:9092 \

--kafka-conf topic=paimon\_canal \

--kafka-conf properties.group.id=atguigu \

--kafka-conf scan.startup.mode=earliest-offset \

--kafka-conf value.format=canal-json \

--catalog-conf metastore=hive \

--catalog-conf uri=thrift://hadoop102:9083 \

--table-conf bucket=4 \

--table-conf changelog-producer=input \

--table-conf sink.parallelism=4

2.8.2.2 同步数据库

**1）语法说明**

将多个主题或一个主题同步到一个 Paimon 数据库中。

<FLINK\_HOME>/bin/flink run \

/path/to/paimon-flink-action-0.5-SNAPSHOT.jar \

kafka-sync-database

--warehouse <warehouse-path> \

--database <database-name> \

[--schema-init-max-read <int>] \

[--ignore-incompatible <true/false>] \

[--table-prefix <paimon-table-prefix>] \

[--table-suffix <paimon-table-suffix>] \

[--including-tables <table-name|name-regular-expr>] \

[--excluding-tables <table-name|name-regular-expr>] \

[--kafka-conf <kafka-source-conf> [--kafka-conf <kafka-source-conf> ...]] \

[--catalog-conf <paimon-catalog-conf> [--catalog-conf <paimon-catalog-conf> ...]] \

[--table-conf <paimon-table-sink-conf> [--table-conf <paimon-table-sink-conf> ...]]

参数说明：

| 配置 | 描述 |
| --- | --- |
| --warehouse | The path to Paimon warehouse.通往派蒙仓库的道路。 |
| --database | Paimon 目录中的数据库名称。 |
| --schema-init-max-read | 如果您的表全部来自某个Topic，您可以设置该参数来初始化需要同步的表数量。默认值为 1000。 |
| --ignore-incompatible | 默认为 false，在这种情况下，如果 Paimon 中存在 MySQL 表名，并且它们的 schema 不兼容，则会抛出异常。您可以显式将其指定为 true 以忽略不兼容的表和异常。 |
| --table-prefix | 所有需要同步的Paimon表的前缀。例如，如果您希望所有同步表都以“ods\_”作为前缀，则可以指定“--table-prefix ods\_”。 |
| --table-suffix | 所有需要同步的Paimon表的后缀。用法与“--table-prefix”相同。 |
| --including-tables | 用于指定要同步哪些源表。您必须使用“|”分隔多个表。因为“|”为特殊字符，需要逗号，例如：'a|b|c'。支持正则表达式，例如指定“--include-tables test|paimon.\*”表示同步表'test'和所有表都以“paimon”开头。 |
| --excluding-tables | 用于指定哪些源表不同步。用法与“--include-tables”相同。如果同时指定了“-- except-tables”，则“-- except-tables”的优先级高于“--include-tables”。 |
| --kafka-conf | Flink Kafka 源的配置。每个配置都应以“key=value”的格式指定。 `properties.bootstrap.servers`、`topic`、`properties.group.id` 和 `value.format` 是必需配置，其他配置是可选的。有关完整配置列表，请参阅其文档。 |
| --catalog-conf | Paimon 目录的配置。每个配置都应以“key=value”的格式指定。请参阅此处以获取目录配置的完整列表。 |
| --table-conf | Paimon 餐桌水槽的配置。每个配置都应以“key=value”的格式指定。请参阅此处了解表配置的完整列表。 |

只有具有主键的表才会被同步。

对于每个要同步的Kafka主题的表，如果对应的Paimon表不存在，该操作将自动创建该表。它的schema将从所有指定的Kafka topic的表中派生出来，它从topic中获取最早的非DDL数据解析schema。如果 Paimon 表已存在，则其schema将与所有指定 Kafka 主题表的schema进行比较。

**2）案例实操**

（1）准备数据（canal-json格式）

为了方便，直接将canal格式的数据插入topic里（user\_info和spu\_info多表数据）：

kafka-console-producer.sh --broker-list hadoop102:9092 --topic paimon\_canal\_2

#插入数据如下(注意不要有空行)：

{"data":[{"id":"6","login\_name":"t7dk2h","nick\_name":"冰冰11","passwd":null,"name":"淳于冰","phone\_num":"13178654378","email":"t7dk2h@263.net","head\_img":null,"user\_level":"1","birthday":"1997-12-08","gender":null,"create\_time":"2022-06-08 00:00:00","operate\_time":null,"status":null}],"database":"gmall","es":1689150607000,"id":1,"isDdl":false,"mysqlType":{"id":"bigint","login\_name":"varchar(200)","nick\_name":"varchar(200)","passwd":"varchar(200)","name":"varchar(200)","phone\_num":"varchar(200)","email":"varchar(200)","head\_img":"varchar(200)","user\_level":"varchar(200)","birthday":"date","gender":"varchar(1)","create\_time":"datetime","operate\_time":"datetime","status":"varchar(200)"},"old":[{"nick\_name":"冰冰"}],"pkNames":["id"],"sql":"","sqlType":{"id":-5,"login\_name":12,"nick\_name":12,"passwd":12,"name":12,"phone\_num":12,"email":12,"head\_img":12,"user\_level":12,"birthday":91,"gender":12,"create\_time":93,"operate\_time":93,"status":12},"table":"user\_info","ts":1689151566836,"type":"UPDATE"}

{"data":[{"id":"7","login\_name":"vihcj30p1","nick\_name":"豪心22","passwd":null,"name":"魏豪心","phone\_num":"13956932645","email":"vihcj30p1@live.com","head\_img":null,"user\_level":"1","birthday":"1991-06-07","gender":"M","create\_time":"2022-06-08 00:00:00","operate\_time":null,"status":null}],"database":"gmall","es":1689151623000,"id":2,"isDdl":false,"mysqlType":{"id":"bigint","login\_name":"varchar(200)","nick\_name":"varchar(200)","passwd":"varchar(200)","name":"varchar(200)","phone\_num":"varchar(200)","email":"varchar(200)","head\_img":"varchar(200)","user\_level":"varchar(200)","birthday":"date","gender":"varchar(1)","create\_time":"datetime","operate\_time":"datetime","status":"varchar(200)"},"old":[{"nick\_name":"豪心"}],"pkNames":["id"],"sql":"","sqlType":{"id":-5,"login\_name":12,"nick\_name":12,"passwd":12,"name":12,"phone\_num":12,"email":12,"head\_img":12,"user\_level":12,"birthday":91,"gender":12,"create\_time":93,"operate\_time":93,"status":12},"table":"user\_info","ts":1689151623139,"type":"UPDATE"}

{"data":[{"id":"8","login\_name":"02r2ahx","nick\_name":"卿卿33","passwd":null,"name":"穆卿","phone\_num":"13412413361","email":"02r2ahx@sohu.com","head\_img":null,"user\_level":"1","birthday":"2001-07-08","gender":"F","create\_time":"2022-06-08 00:00:00","operate\_time":null,"status":null}],"database":"gmall","es":1689151626000,"id":3,"isDdl":false,"mysqlType":{"id":"bigint","login\_name":"varchar(200)","nick\_name":"varchar(200)","passwd":"varchar(200)","name":"varchar(200)","phone\_num":"varchar(200)","email":"varchar(200)","head\_img":"varchar(200)","user\_level":"varchar(200)","birthday":"date","gender":"varchar(1)","create\_time":"datetime","operate\_time":"datetime","status":"varchar(200)"},"old":[{"nick\_name":"卿卿"}],"pkNames":["id"],"sql":"","sqlType":{"id":-5,"login\_name":12,"nick\_name":12,"passwd":12,"name":12,"phone\_num":12,"email":12,"head\_img":12,"user\_level":12,"birthday":91,"gender":12,"create\_time":93,"operate\_time":93,"status":12},"table":"user\_info","ts":1689151626863,"type":"UPDATE"}

{"data":[{"id":"9","login\_name":"mjhrxnu","nick\_name":"武新44","passwd":null,"name":"罗武新","phone\_num":"13617856358","email":"mjhrxnu@yahoo.com","head\_img":null,"user\_level":"1","birthday":"2001-08-08","gender":null,"create\_time":"2022-06-08 00:00:00","operate\_time":null,"status":null}],"database":"gmall","es":1689151630000,"id":4,"isDdl":false,"mysqlType":{"id":"bigint","login\_name":"varchar(200)","nick\_name":"varchar(200)","passwd":"varchar(200)","name":"varchar(200)","phone\_num":"varchar(200)","email":"varchar(200)","head\_img":"varchar(200)","user\_level":"varchar(200)","birthday":"date","gender":"varchar(1)","create\_time":"datetime","operate\_time":"datetime","status":"varchar(200)"},"old":[{"nick\_name":"武新"}],"pkNames":["id"],"sql":"","sqlType":{"id":-5,"login\_name":12,"nick\_name":12,"passwd":12,"name":12,"phone\_num":12,"email":12,"head\_img":12,"user\_level":12,"birthday":91,"gender":12,"create\_time":93,"operate\_time":93,"status":12},"table":"user\_info","ts":1689151630781,"type":"UPDATE"}

{"data":[{"id":"10","login\_name":"kwua2155","nick\_name":"纨纨55","passwd":null,"name":"姜纨","phone\_num":"13742843828","email":"kwua2155@163.net","head\_img":null,"user\_level":"3","birthday":"1997-11-08","gender":"F","create\_time":"2022-06-08 00:00:00","operate\_time":null,"status":null}],"database":"gmall","es":1689151633000,"id":5,"isDdl":false,"mysqlType":{"id":"bigint","login\_name":"varchar(200)","nick\_name":"varchar(200)","passwd":"varchar(200)","name":"varchar(200)","phone\_num":"varchar(200)","email":"varchar(200)","head\_img":"varchar(200)","user\_level":"varchar(200)","birthday":"date","gender":"varchar(1)","create\_time":"datetime","operate\_time":"datetime","status":"varchar(200)"},"old":[{"nick\_name":"纨纨"}],"pkNames":["id"],"sql":"","sqlType":{"id":-5,"login\_name":12,"nick\_name":12,"passwd":12,"name":12,"phone\_num":12,"email":12,"head\_img":12,"user\_level":12,"birthday":91,"gender":12,"create\_time":93,"operate\_time":93,"status":12},"table":"user\_info","ts":1689151633697,"type":"UPDATE"}

{"data":[{"id":"12","spu\_name":"华为智慧屏 4K全面屏智能电视机1","description":"华为智慧屏 4K全面屏智能电视机","category3\_id":"86","tm\_id":"3","create\_time":"2021-12-14 00:00:00","operate\_time":null}],"database":"gmall","es":1689151648000,"id":6,"isDdl":false,"mysqlType":{"id":"bigint","spu\_name":"varchar(200)","description":"varchar(1000)","category3\_id":"bigint","tm\_id":"bigint","create\_time":"datetime","operate\_time":"datetime"},"old":[{"spu\_name":"华为智慧屏 4K全面屏智能电视机"}],"pkNames":["id"],"sql":"","sqlType":{"id":-5,"spu\_name":12,"description":12,"category3\_id":-5,"tm\_id":-5,"create\_time":93,"operate\_time":93},"table":"spu\_info","ts":1689151648872,"type":"UPDATE"}

{"data":[{"id":"3","spu\_name":"Apple iPhone 13","description":"Apple iPhone 13","category3\_id":"61","tm\_id":"2","create\_time":"2021-12-14 00:00:00","operate\_time":null}],"database":"gmall","es":1689151661000,"id":7,"isDdl":false,"mysqlType":{"id":"bigint","spu\_name":"varchar(200)","description":"varchar(1000)","category3\_id":"bigint","tm\_id":"bigint","create\_time":"datetime","operate\_time":"datetime"},"old":[{"spu\_name":"Apple iPhone 12","description":"Apple iPhone 12"}],"pkNames":["id"],"sql":"","sqlType":{"id":-5,"spu\_name":12,"description":12,"category3\_id":-5,"tm\_id":-5,"create\_time":93,"operate\_time":93},"table":"spu\_info","ts":1689151661828,"type":"UPDATE"}

{"data":[{"id":"4","spu\_name":"HUAWEI P50","description":"HUAWEI P50","category3\_id":"61","tm\_id":"3","create\_time":"2021-12-14 00:00:00","operate\_time":null}],"database":"gmall","es":1689151669000,"id":8,"isDdl":false,"mysqlType":{"id":"bigint","spu\_name":"varchar(200)","description":"varchar(1000)","category3\_id":"bigint","tm\_id":"bigint","create\_time":"datetime","operate\_time":"datetime"},"old":[{"spu\_name":"HUAWEI P40","description":"HUAWEI P40"}],"pkNames":["id"],"sql":"","sqlType":{"id":-5,"spu\_name":12,"description":12,"category3\_id":-5,"tm\_id":-5,"create\_time":93,"operate\_time":93},"table":"spu\_info","ts":1689151669966,"type":"UPDATE"}

{"data":[{"id":"1","spu\_name":"小米12sultra","description":"小米12","category3\_id":"61","tm\_id":"1","create\_time":"2021-12-14 00:00:00","operate\_time":null}],"database":"gmall","es":1689151700000,"id":9,"isDdl":false,"mysqlType":{"id":"bigint","spu\_name":"varchar(200)","description":"varchar(1000)","category3\_id":"bigint","tm\_id":"bigint","create\_time":"datetime","operate\_time":"datetime"},"old":[{"description":"小米10"}],"pkNames":["id"],"sql":"","sqlType":{"id":-5,"spu\_name":12,"description":12,"category3\_id":-5,"tm\_id":-5,"create\_time":93,"operate\_time":93},"table":"spu\_info","ts":1689151700998,"type":"UPDATE"}

再准备一个只包含spu\_info单表数据的Topic：

kafka-console-producer.sh --broker-list hadoop102:9092 --topic paimon\_canal\_1

#插入数据如下：

{"data":[{"id":"12","spu\_name":"华为智慧屏 4K全面屏智能电视机1","description":"华为智慧屏 4K全面屏智能电视机","category3\_id":"86","tm\_id":"3","create\_time":"2021-12-14 00:00:00","operate\_time":null}],"database":"gmall","es":1689151648000,"id":6,"isDdl":false,"mysqlType":{"id":"bigint","spu\_name":"varchar(200)","description":"varchar(1000)","category3\_id":"bigint","tm\_id":"bigint","create\_time":"datetime","operate\_time":"datetime"},"old":[{"spu\_name":"华为智慧屏 4K全面屏智能电视机"}],"pkNames":["id"],"sql":"","sqlType":{"id":-5,"spu\_name":12,"description":12,"category3\_id":-5,"tm\_id":-5,"create\_time":93,"operate\_time":93},"table":"spu\_info","ts":1689151648872,"type":"UPDATE"}

{"data":[{"id":"3","spu\_name":"Apple iPhone 13","description":"Apple iPhone 13","category3\_id":"61","tm\_id":"2","create\_time":"2021-12-14 00:00:00","operate\_time":null}],"database":"gmall","es":1689151661000,"id":7,"isDdl":false,"mysqlType":{"id":"bigint","spu\_name":"varchar(200)","description":"varchar(1000)","category3\_id":"bigint","tm\_id":"bigint","create\_time":"datetime","operate\_time":"datetime"},"old":[{"spu\_name":"Apple iPhone 12","description":"Apple iPhone 12"}],"pkNames":["id"],"sql":"","sqlType":{"id":-5,"spu\_name":12,"description":12,"category3\_id":-5,"tm\_id":-5,"create\_time":93,"operate\_time":93},"table":"spu\_info","ts":1689151661828,"type":"UPDATE"}

{"data":[{"id":"4","spu\_name":"HUAWEI P50","description":"HUAWEI P50","category3\_id":"61","tm\_id":"3","create\_time":"2021-12-14 00:00:00","operate\_time":null}],"database":"gmall","es":1689151669000,"id":8,"isDdl":false,"mysqlType":{"id":"bigint","spu\_name":"varchar(200)","description":"varchar(1000)","category3\_id":"bigint","tm\_id":"bigint","create\_time":"datetime","operate\_time":"datetime"},"old":[{"spu\_name":"HUAWEI P40","description":"HUAWEI P40"}],"pkNames":["id"],"sql":"","sqlType":{"id":-5,"spu\_name":12,"description":12,"category3\_id":-5,"tm\_id":-5,"create\_time":93,"operate\_time":93},"table":"spu\_info","ts":1689151669966,"type":"UPDATE"}

{"data":[{"id":"1","spu\_name":"小米12sultra","description":"小米12","category3\_id":"61","tm\_id":"1","create\_time":"2021-12-14 00:00:00","operate\_time":null}],"database":"gmall","es":1689151700000,"id":9,"isDdl":false,"mysqlType":{"id":"bigint","spu\_name":"varchar(200)","description":"varchar(1000)","category3\_id":"bigint","tm\_id":"bigint","create\_time":"datetime","operate\_time":"datetime"},"old":[{"description":"小米10"}],"pkNames":["id"],"sql":"","sqlType":{"id":-5,"spu\_name":12,"description":12,"category3\_id":-5,"tm\_id":-5,"create\_time":93,"operate\_time":93},"table":"spu\_info","ts":1689151700998,"type":"UPDATE"}

（2）从一个 Kafka 主题（包含多表数据）同步到 Paimon 数据库

bin/flink run \

/opt/module/flink-1.17.0/opt/paimon-flink-action-0.5-20230703.002437-53.jar \

kafka-sync-database \

--warehouse hdfs://hadoop102:8020/paimon/hive \

--database test \

--table-prefix "t1\_" \

--table-suffix "\_cdc" \

--schema-init-max-read 500 \

--kafka-conf properties.bootstrap.servers=hadoop102:9092 \

--kafka-conf topic=paimon\_canal\_2 \

--kafka-conf properties.group.id=atguigu \

--kafka-conf scan.startup.mode=earliest-offset \

--kafka-conf value.format=canal-json \

--catalog-conf metastore=hive \

--catalog-conf uri=thrift://hadoop102:9083 \

--table-conf bucket=4 \

--table-conf changelog-producer=input \

--table-conf sink.parallelism=4

从多个 Kafka 主题同步到 Paimon 数据库

bin/flink run \

/opt/module/flink-1.17.0/opt/paimon-flink-action-0.5-20230703.002437-53.jar \

kafka-sync-database \

--warehouse hdfs://hadoop102:8020/paimon/hive \

--database test \

--table-prefix "t2\_" \

--table-suffix "\_cdc" \

--kafka-conf properties.bootstrap.servers=hadoop102:9092 \

--kafka-conf topic="paimon\_canal;paimon\_canal\_1" \

--kafka-conf properties.group.id=atguigu \

--kafka-conf scan.startup.mode=earliest-offset \

--kafka-conf value.format=canal-json \

--catalog-conf metastore=hive \

--catalog-conf uri=thrift://hadoop102:9083 \

--table-conf bucket=4 \

--table-conf changelog-producer=input \

--table-conf sink.parallelism=4

### 2.8.3 支持的schema变更

cdc 集成支持有限的schema变更。目前，框架无法删除列，因此 DROP 的行为将被忽略，RENAME 将添加新列。当前支持的架构更改包括：

（1）添加列。

（2）更改列类型：

* 从字符串类型（char、varchar、text）更改为长度更长的另一种字符串类型，
* 从二进制类型（binary、varbinary、blob）更改为长度更长的另一种二进制类型，
* 从整数类型（tinyint、smallint、int、bigint）更改为范围更广的另一种整数类型，
* 从浮点类型（float、double）更改为范围更宽的另一种浮点类型。

## 2.9 进阶使用

### 2.9.1 写入性能

Paimon的写入性能与检查点密切相关，因此需要更大的写入吞吐量：

* 增加检查点间隔，或者仅使用批处理模式。
* 增加写入缓冲区大小。
* 启用写缓冲区溢出。
* 如果您使用固定存储桶模式，请重新调整存储桶数量。

2.9.1.1 并行度

建议sink的并行度小于等于bucket的数量，最好相等。

| **选项** | **必需的** | **默认** | **类型** | **描述** |
| --- | --- | --- | --- | --- |
| sink.parallelism | No | (none) | Integer | 定义sink的并行度。默认情况下，并行度由框架使用上游链式运算符的相同并行度来确定。 |

2.9.1.2 Compaction

当Sorted Run数量较少时，Paimon writer 将在单独的线程中异步执行压缩，因此记录可以连续写入表中。然而，为了避免Sorted Runs的无限增长，当Sorted Run的数量达到阈值时，writer将不得不暂停写入。下表属性确定阈值。

| **选项** | **必需的** | **默认** | **类型** | **描述** |
| --- | --- | --- | --- | --- |
| num-sorted-run.stop-trigger | No | (none) | Integer | 触发停止写入的Sorted Runs次数，默认值为 'num-sorted-run.compaction-trigger' + 1。 |

当 num-sorted-run.stop-trigger 变大时，写入停顿将变得不那么频繁，从而提高写入性能。但是，如果该值变得太大，则查询表时将需要更多内存和 CPU 时间。如果您担心内存 OOM，请配置sort-spill-threshold。它的值取决于你的内存大小。

2.9.1.3 优先考虑写入吞吐量

如果希望某种模式具有最大写入吞吐量，则可以缓慢而不是匆忙地进行Compaction。可以对表使用以下策略

num-sorted-run.stop-trigger = 2147483647

sort-spill-threshold = 10

此配置将在写入高峰期生成更多文件，并在写入低谷期逐渐合并到最佳读取性能。

2.9.1.4 触发Compaction的Sorted Run数

Paimon使用LSM树，支持大量更新。 LSM 在多次Sorted Runs中组织文件。从 LSM 树查询记录时，必须组合所有Sorted Runs以生成所有记录的完整视图。

过多的Sorted Run会导致查询性能不佳。为了将Sorted Run的数量保持在合理的范围内，Paimon writers 将自动执行Compaction。下表属性确定触发Compaction的最小Sorted Run数。

| **选项** | **必需的** | **默认** | **类型** | **描述** |
| --- | --- | --- | --- | --- |
| num-sorted-run.compaction-trigger | No | 5 | Integer | 触发Compaction的Sorted Run数。包括 0 级文件（一个文件一级排序运行）和高级运行（一个一级排序运行）。 |

2.9.1.5 写入初始化

在write初始化时，bucket的writer需要读取所有历史文件。如果这里出现瓶颈（例如同时写入大量分区），可以使用write-manifest-cache缓存读取的manifest数据，以加速初始化。

2.9.1.6 内存

Paimon writer中主要占用内存的地方有3个：

* Writer的内存缓冲区，由单个任务的所有Writer共享和抢占。该内存值可以通过 write-buffer-size 表属性进行调整。
* 合并多个Sorted Run以进行Compaction时会消耗内存。可以通过 num-sorted-run.compaction-trigger 选项进行调整，以更改要合并的Sorted Run的数量。
* 如果行非常大，在进行Compaction时一次读取太多行数据可能会消耗大量内存。减少 read.batch-size 选项可以减轻这种情况的影响。
* 写入列式（ORC、Parquet等）文件所消耗的内存，不可调。

### 2.9.2 读取性能

2.9.2.1 Full Compaction

配置“full-compaction.delta-commits”在Flink写入中定期执行full-compaction。并且可以确保在写入结束之前分区被完全Compaction。

注意：Paimon 默认处理小文件并提供良好的读取性能。请不要在没有任何要求的情况下配置此Full Compaction选项，因为它会对性能产生重大影响。

2.9.2.2 主键表

对于主键表来说，这是一种“MergeOnRead”技术。读取数据时，会合并多层LSM数据，并行数会受到桶数的限制。虽然Paimon的merge会高效，但是还是赶不上普通的AppendOnly表。

如果你想在某些场景下查询得足够快，但只能找到较旧的数据，你可以：

* 配置full-compaction.delta-commits，写入数据时（目前只有Flink）会定期进行full Compaction。
* 配置“scan.mode”为“compacted-full”，读取数据时，选择full-compaction的快照。读取性能良好。

2.9.2.3 仅追加表

小文件会降低读取速度并影响 DFS 稳定性。默认情况下，当单个存储桶中的小文件超过“compaction.max.file-num”（默认50个）时，就会触发compaction。但是当有多个桶时，就会产生很多小文件。

您可以使用full-compaction来减少小文件。full-compaction将消除大多数小文件。

2.9.2.4 格式

Paimon 对 parquet 读取进行了一些查询优化，因此 parquet 会比 orc 稍快一些。

### 2.9.3 多Writer并发写入

Paimon的快照管理支持向多个writer写入。

默认情况下，Paimon支持对不同分区的并发写入。推荐的方式是streaming job将记录写入Paimon的最新分区；同时批处理作业（覆盖）将记录写入历史分区。

图示

描述已自动生成

如果需要多个Writer写到同一个分区，事情就会变得有点复杂。例如，不想使用 UNION ALL，那就需要有多个流作业来写入“partial-update”表。参考如下的“Dedicated Compaction Job”。

2.9.3.1 Dedicated Compaction Job

默认情况下，Paimon writer 在写入记录时会根据需要执行Compaction。这对于大多数用例来说已经足够了，但有两个缺点：

* 这可能会导致写入吞吐量不稳定，因为执行压缩时吞吐量可能会暂时下降。
* Compaction会将某些数据文件标记为“已删除”（并未真正删除）。如果多个writer标记同一个文件，则在提交更改时会发生冲突。 Paimon 会自动解决冲突，但这可能会导致作业重新启动。

为了避免这些缺点，用户还可以选择在writer中跳过Compaction，并仅运行专门的作业来进行Compaction。由于Compaction仅由专用作业执行，因此writer可以连续写入记录而无需暂停，并且不会发生冲突。

| **选项** | **必需的** | **默认** | **类型** | **描述** |
| --- | --- | --- | --- | --- |
| write-only | No | false | Boolean | 如果设置为 true，将跳过Compaction和快照过期。此选项与独立Compaction一起使用。 |

Flink SQL目前不支持compaction相关的语句，所以我们必须通过flink run来提交compaction作业。

<FLINK\_HOME>/bin/flink run \

/path/to/paimon-flink-action-0.5-SNAPSHOT.jar \

compact \

--warehouse <warehouse-path> \

--database <database-name> \

--table <table-name> \

[--partition <partition-name>] \

[--catalog-conf <paimon-catalog-conf> [--catalog-conf <paimon-catalog-conf> ...]] \

如果提交一个批处理作业（execution.runtime-mode:batch），当前所有的表文件都会被Compaction。如果您提交一个流作业（execution.runtime-mode: Streaming），该作业将持续监视表的新更改并根据需要执行Compaction。

### 2.9.4 表管理

2.9.4.1 管理快照

**1）快照过期**

Paimon Writer每次提交都会生成一个或两个快照。每个快照可能会添加一些新的数据文件或将一些旧的数据文件标记为已删除。然而，标记的数据文件并没有真正被删除，因为Paimon还支持时间旅行到更早的快照。它们仅在快照过期时被删除。

目前，Paimon Writer在提交新更改时会自动执行过期操作。通过使旧快照过期，可以删除不再使用的旧数据文件和元数据文件，以释放磁盘空间。

设置以下表属性：

| **选项** | **必需的** | **默认** | **类型** | **描述** |
| --- | --- | --- | --- | --- |
| snapshot.time-retained | No | 1 h | Duration | 已完成快照的最长时间保留。 |
| snapshot.num-retained.min | No | 10 | Integer | 要保留的已完成快照的最小数量。 |
| snapshot.num-retained.max | No | Integer.MAX\_VALUE | Integer | 要保留的已完成快照的最大数量。 |

注意，保留时间太短或保留数量太少可能会导致如下问题：

* 批量查询找不到该文件。例如，表比较大，批量查询需要10分钟才能读取，但是10分钟前的快照过期了，此时批量查询会读取到已删除的快照。
* 表文件上的流式读取作业（没有外部日志系统）无法重新启动。当作业重新启动时，它记录的快照可能已过期。 （可以使用Consumer Id来保护快照过期的小保留时间内的流式读取）。

**2）回滚快照**

<FLINK\_HOME>/bin/flink run \

/path/to/paimon-flink-action-0.5-SNAPSHOT.jar \

rollback-to \

--warehouse <warehouse-path> \

--database <database-name> \

--table <table-name> \

--snapshot <snapshot-id> \

[--catalog-conf <paimon-catalog-conf> [--catalog-conf <paimon-catalog-conf> ...]]

2.9.4.2 管理分区

创建分区表时可以设置partition.expiration-time。 Paimon会定期检查分区的状态，并根据时间删除过期的分区。

判断分区是否过期：将分区中提取的时间与当前时间进行比较，看生存时间是否超过partition.expiration-time。比如：

CREATE TABLE T (...) PARTITIONED BY (dt) WITH (

'partition.expiration-time' = '7 d',

'partition.expiration-check-interval' = '1 d',

'partition.timestamp-formatter' = 'yyyyMMdd'

);

| **选项** | **默认** | **类型** | **描述** |
| --- | --- | --- | --- |
| partition.expiration-check-interval | 1 h | Duration | 分区过期的检查间隔。 |
| partition.expiration-time | (none) | Duration | 分区的过期时间间隔。如果分区的生命周期超过此值，则该分区将过期。分区时间是从分区值中提取的。 |
| partition.timestamp-formatter | (none) | String | 用于格式化字符串时间戳的格式化程序。它可以与“partition.timestamp-pattern”一起使用来创建使用指定值的格式化程序。   * 默认格式化程序为“yyyy-MM-dd HH:mm:ss”和“yyyy-MM-dd”。 * 支持多个分区字段，例如“$year-$month-$day $hour:00:00”。 * 时间戳格式化程序与 Java 的 DateTimeFormatter 兼容。 |
| partition.timestamp-pattern | (none) | String | 可以指定一种模式来从分区获取时间戳。格式化程序模式由“partition.timestamp-formatter”定义。   * 默认情况下，从第一个字段读取。 * 如果分区中的时间戳是名为“dt”的单个字段，则可以使用“$dt”。 * 如果它分布在年、月、日和小时的多个字段中，则可以使用“$year-$month-$day $hour:00:00”。 * 如果时间戳位于 dt 和 hour 字段中，则可以使用“$dt $hour:00:00”。 |

2.9.4.3 管理小文件

小文件可能会导致：

* 稳定性问题：HDFS中小文件过多，NameNode会承受过大的压力。
* 成本问题：HDFS中的小文件会暂时使用最小1个Block的大小，例如128MB。
* 查询效率：小文件过多查询效率会受到影响。

**1）Flink Checkpoint的影响**

使用Flink Writer，每个checkpoint会生成 1-2 个快照，并且checkpoint会强制在 DFS 上生成文件，因此checkpoint间隔越小，会生成越多的小文件。

默认情况下，不仅checkpoint会导致文件生成，writer的内存（write-buffer-size）耗尽也会将数据flush到DFS并生成相应的文件。可以启用 write-buffer-spillable 在 writer 中生成溢出文件，从而在 DFS 中生成更大的文件。

所以，可以设置如下：

* 增大checkpoint间隔
* 增加 write-buffer-size 或启用 write-buffer-spillable

**2）快照的影响**

Paimon维护文件的多个版本，文件的Compaction和删除是逻辑上的，并没有真正删除文件。文件只有在 Snapshot 过期后才会被真正删除，因此减少文件的第一个方法就是减少 Snapshot 过期的时间。 Flink writer 会自动使快照过期。

**分区和分桶的影响**

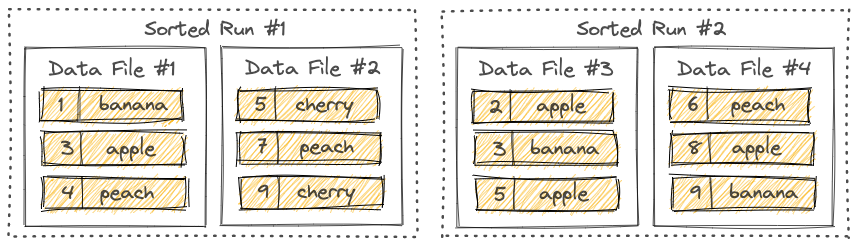
图示

描述已自动生成

表数据会被物理分片到不同的分区，里面有不同的桶，所以如果整体数据量太小，单个桶中至少有一个文件，建议你配置较少的桶数，否则会出现也有很多小文件。

3）**主键表LSM的影响**

LSM 树将文件组织成Sorted Runs的运行。Sorted Runs由一个或多个数据文件组成，并且每个数据文件恰好属于一个Sorted Runs。



默认情况下，Sorted Runs数取决于 num-sorted-run.compaction-trigger，这意味着一个桶中至少有 5 个文件。如果要减少此数量，可以保留更少的文件，但写入性能可能会受到影响。

**4）仅追加表的文件的影响**

默认情况下，Append-Only 还会进行自动Compaction以减少小文件的数量

对于分桶的 Append-only 表，为了排序会对bucket内的文件行Compaction，可能会保留更多的小文件。

**5）Full-Compaction的影响**

主键表是5个文件，但是Append-Only表（桶）可能单个桶里有50个小文件，这是很难接受的。更糟糕的是，不再活动的分区还保留了如此多的小文件。

建议配置Full-Compaction，在Flink写入时配置‘full-compaction.delta-commits’定期进行full-compaction。并且可以确保在写入结束之前分区被full-compaction。

### 2.9.5 缩放Bucket

**1）说明**

由于总桶数对性能影响很大，Paimon 允许用户通过 ALTER TABLE 命令调整桶数，并通过 INSERT OVERWRITE 重新组织数据布局，而无需重新创建表/分区。当执行覆盖作业时，框架会自动扫描旧桶号的数据，并根据当前桶号对记录进行哈希处理。

-- rescale number of total buckets

ALTER TABLE table\_identifier SET ('bucket' = '...')

-- reorganize data layout of table/partition

INSERT OVERWRITE table\_identifier [PARTITION (part\_spec)]

SELECT ...

FROM table\_identifier

[WHERE part\_spec]

注意：

* ALTER TABLE 仅修改表的元数据，不会重新组织或重新格式化现有数据。重新组织现有数据必须通过INSERT OVERWRITE来实现。
* 重新缩放桶数不会影响读取和正在运行的写入作业。
* 一旦存储桶编号更改，任何新安排的 INSERT INTO 作业写入未重新组织的现有表/分区将抛出 TableException ，并显示如下类似异常：

Try to write table/partition ... with a new bucket num ...,

but the previous bucket num is ... Please switch to batch mode,

and perform INSERT OVERWRITE to rescale current data layout first.

对于分区表，不同的分区可以有不同的桶号。例如：

ALTER TABLE my\_table SET ('bucket' = '4');

INSERT OVERWRITE my\_table PARTITION (dt = '2022-01-01')

SELECT \* FROM ...;

ALTER TABLE my\_table SET ('bucket' = '8');

INSERT OVERWRITE my\_table PARTITION (dt = '2022-01-02')

SELECT \* FROM ...;

在覆盖期间，确保没有其他作业写入同一表/分区。

注意：对于启用日志系统的表（例如Kafka），请重新调整主题的分区以保持一致性。

重新缩放存储桶有助于处理吞吐量的突然峰值。假设有一个每日流式ETL任务来同步交易数据。该表的DDL和管道如下所示。

**2）官方示例：**

如下是正在跑的一个作业：

-- 建表

CREATE TABLE verified\_orders (

trade\_order\_id BIGINT,

item\_id BIGINT,

item\_price DOUBLE,

dt STRING,

PRIMARY KEY (dt, trade\_order\_id, item\_id) NOT ENFORCED

) PARTITIONED BY (dt)

WITH (

'bucket' = '16'

);

-- kafka表

CREATE temporary TABLE raw\_orders(

trade\_order\_id BIGINT,

item\_id BIGINT,

item\_price BIGINT,

gmt\_create STRING,

order\_status STRING

) WITH (

'connector' = 'kafka',

'topic' = '...',

'properties.bootstrap.servers' = '...',

'format' = 'csv'

...

);

-- 流式插入16个分桶

INSERT INTO verified\_orders

SELECT trade\_order\_id,

item\_id,

item\_price,

DATE\_FORMAT(gmt\_create, 'yyyy-MM-dd') AS dt

FROM raw\_orders

WHERE order\_status = 'verified';

过去几周运行良好。然而，最近数据量增长很快，作业的延迟不断增加。为了提高数据新鲜度，用户可以执行如下操作缩放分桶：

（1）使用保存点暂停流作业

$ ./bin/flink stop \

--savepointPath /tmp/flink-savepoints \

$JOB\_ID

（2）增加桶数

ALTER TABLE verified\_orders SET ('bucket' = '32');

（3）切换到批处理模式并覆盖流作业正在写入的当前分区

SET 'execution.runtime-mode' = 'batch';

-- 假设今天是2022-06-22

-- 情况1：没有更新历史分区的延迟事件，因此覆盖今天的分区就足够了

INSERT OVERWRITE verified\_orders PARTITION (dt = '2022-06-22')

SELECT trade\_order\_id,

item\_id,

item\_price

FROM verified\_orders

WHERE dt = '2022-06-22';

- 情况2：有更新历史分区的延迟事件，但范围不超过3天

INSERT OVERWRITE verified\_orders

SELECT trade\_order\_id,

item\_id,

item\_price,

dt

FROM verified\_orders

WHERE dt IN ('2022-06-20', '2022-06-21', '2022-06-22');

（4）覆盖作业完成后，切换回流模式，从保存点恢复（可以增加并行度=新bucket数量）。

SET 'execution.runtime-mode' = 'streaming';

SET 'execution.savepoint.path' = <savepointPath>;

INSERT INTO verified\_orders

SELECT trade\_order\_id,

item\_id,

item\_price,

DATE\_FORMAT(gmt\_create, 'yyyy-MM-dd') AS dt

FROM raw\_orders

WHERE order\_status = 'verified';

## 2.10 文件操作理解

### 2.10.1 插入数据

当我们执行INSERT INTO

CREATE CATALOG paimon WITH (

'type' = 'paimon',

'warehouse' = 'file:///tmp/paimon'

);

USE CATALOG paimon;

CREATE TABLE T (

id BIGINT,

a INT,

b STRING,

dt STRING COMMENT 'timestamp string in format yyyyMMdd',

PRIMARY KEY(id, dt) NOT ENFORCED

) PARTITIONED BY (dt);

INSERT INTO T VALUES (1, 10001, 'varchar00001', '20230501');

一旦Flink作业完成，记录就会通过成功提交写入Paimon表中。用户可以通过执行查询 SELECT \* FROM T 来验证这些记录的可见性，该查询将返回单行。提交过程创建位于路径 /tmp/paimon/default.db/T/snapshot/snapshot-1 的快照。 snapshot-1 处生成的文件布局如下所述：

信件

描述已自动生成

snapshot-1 的内容包含快照的元数据，例如清单列表（manifest list）和schema ID：

{

"version" : 3,

"id" : 1,

"schemaId" : 0,

"baseManifestList" : "manifest-list-4ccc-c07f-4090-958c-cfe3ce3889e5-0",

"deltaManifestList" : "manifest-list-4ccc-c07f-4090-958c-cfe3ce3889e5-1",

"changelogManifestList" : null,

"commitUser" : "7d758485-981d-4b1a-a0c6-d34c3eb254bf",

"commitIdentifier" : 9223372036854775807,

"commitKind" : "APPEND",

"timeMillis" : 1684155393354,

"logOffsets" : { },

"totalRecordCount" : 1,

"deltaRecordCount" : 1,

"changelogRecordCount" : 0,

"watermark" : -9223372036854775808

}

清单列表包含快照的所有更改，baseManifestList 是应用 deltaManifestList 中的更改的基础文件。第一次提交将生成 1 个清单文件（manifest file），并创建 2 个清单列表（manifest list）：

./T/manifest:

--deltaManifestList：包含对数据文件执行操作的清单条目列表（上图中的 manifest-list-1-delta）

manifest-list-4ccc-c07f-4090-958c-cfe3ce3889e5-1

--baseManifestList：空的（上图中的 manifest-list-1-base）

manifest-list-4ccc-c07f-4090-958c-cfe3ce3889e5-0

--清单文件：存储快照中数据文件的信息（上图中的manifest-1-0）

manifest-2b833ea4-d7dc-4de0-ae0d-ad76eced75cc-0

跨不同分区插入一批记录：

INSERT INTO T VALUES

(2, 10002, 'varchar00002', '20230502'),

(3, 10003, 'varchar00003', '20230503'),

(4, 10004, 'varchar00004', '20230504'),

(5, 10005, 'varchar00005', '20230505'),

(6, 10006, 'varchar00006', '20230506'),

(7, 10007, 'varchar00007', '20230507'),

(8, 10008, 'varchar00008', '20230508'),

(9, 10009, 'varchar00009', '20230509'),

(10, 10010, 'varchar00010', '20230510');

第二次提交发生，执行 SELECT \* FROM T 将返回 10 行。创建一个新快照，即 snapshot-2，并为我们提供以下物理文件布局：

% ls -atR .

./T:

dt=20230501

dt=20230502

dt=20230503

dt=20230504

dt=20230505

dt=20230506

dt=20230507

dt=20230508

dt=20230509

dt=20230510

snapshot

schema

manifest

./T/snapshot:

LATEST

snapshot-2

EARLIEST

snapshot-1

./T/manifest:

manifest-list-9ac2-5e79-4978-a3bc-86c25f1a303f-1 # delta manifest list for snapshot-2

manifest-list-9ac2-5e79-4978-a3bc-86c25f1a303f-0 # base manifest list for snapshot-2

manifest-f1267033-e246-4470-a54c-5c27fdbdd074-0 # manifest file for snapshot-2

manifest-list-4ccc-c07f-4090-958c-cfe3ce3889e5-1 # delta manifest list for snapshot-1

manifest-list-4ccc-c07f-4090-958c-cfe3ce3889e5-0 # base manifest list for snapshot-1

manifest-2b833ea4-d7dc-4de0-ae0d-ad76eced75cc-0 # manifest file for snapshot-1

./T/dt=20230501/bucket-0:

data-b75b7381-7c8b-430f-b7e5-a204cb65843c-0.orc

...

# each partition has the data written to bucket-0

...

./T/schema:

schema-0

截至 snapshot-2 的新文件布局如下所示：

图示, 工程绘图

描述已自动生成

### 2.10.2 删除数据

执行如下删除：

DELETE FROM T WHERE dt >= '20230503';

第三次提交发生，它为我们提供了 snapshot-3。现在，列出表下的文件，您会发现没有分区被删除。相反，会为分区 20230503 到 20230510 创建一个新的数据文件：

./T/dt=20230510/bucket-0:

data-b93f468c-b56f-4a93-adc4-b250b3aa3462-0.orc # newer data file created by the delete statement

data-0fcacc70-a0cb-4976-8c88-73e92769a762-0.orc # older data file created by the insert statement

因为我们在第二次提交中插入一条记录（由 +I[10, 10010, 'varchar00010', '20230510'] 表示），然后在第三次提交中删除该记录。执行 SELECT \* FROM T 将返回 2 行，即：

+I[1, 10001, 'varchar00001', '20230501']

+I[2, 10002, 'varchar00002', '20230502']

截至 snapshot-3 的新文件布局如下所示

图示

描述已自动生成

manifest-3-0包含8个ADD操作类型的manifest条目，对应8个新写入的数据文件。

### 2.10.3 Compaction

小文件的数量会随着连续快照的增加而增加，这可能会导致读取性能下降。因此，需要进行full compaction以减少小文件的数量。

现在触发full-compaction：

./bin/flink run \

./lib/paimon-flink-action-0.5-SNAPSHOT.jar \

compact \

--path file:///tmp/paimon/default.db/T

所有当前表文件将被压缩，并创建一个新快照，即 snapshot-4，并包含以下信息：

{

"version" : 3,

"id" : 4,

"schemaId" : 0,

"baseManifestList" : "manifest-list-9be16-82e7-4941-8b0a-7ce1c1d0fa6d-0",

"deltaManifestList" : "manifest-list-9be16-82e7-4941-8b0a-7ce1c1d0fa6d-1",

"changelogManifestList" : null,

"commitUser" : "a3d951d5-aa0e-4071-a5d4-4c72a4233d48",

"commitIdentifier" : 9223372036854775807,

"commitKind" : "COMPACT",

"timeMillis" : 1684163217960,

"logOffsets" : { },

"totalRecordCount" : 38,

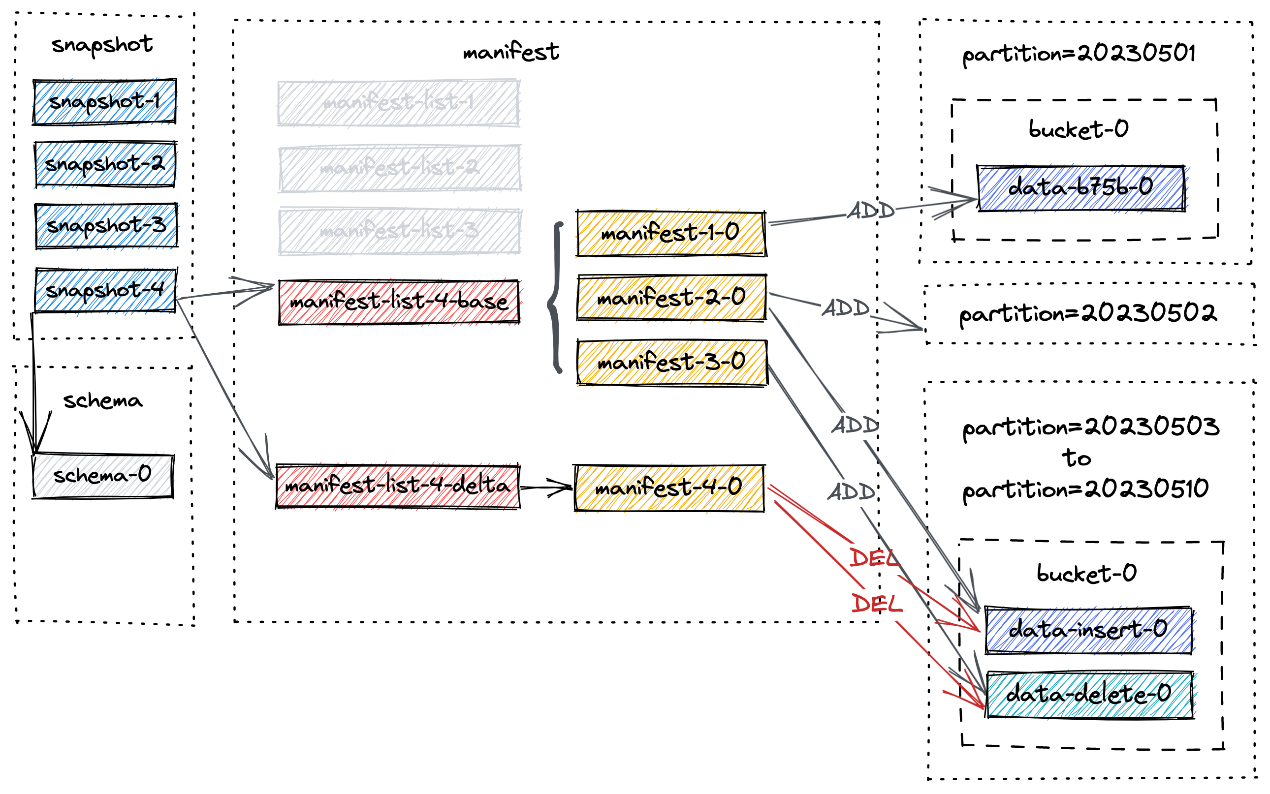
"deltaRecordCount" : 20,

"changelogRecordCount" : 0,

"watermark" : -9223372036854775808

}

截至 snapshot-4 的新文件布局如下所示



manifest-4-0 包含 20 个清单条目（18 个 DELETE 操作和 2 个 ADD 操作）：

* 对于分区20230503到20230510，对两个数据文件进行两次DELETE操作
* 对于分区20230501到20230502，对同一个数据文件进行1次DELETE操作和1次ADD操作。

### 2.10.4 修改表

执行以下语句来配置full-compaction：

ALTER TABLE T SET ('full-compaction.delta-commits' = '1');

它将为 Paimon 表创建一个新schema，即 schema-1，但在下一次提交之前还没有快照实际使用该schema。

### 2.10.5 过期快照

在快照过期的过程中，首先确定快照的范围，然后将这些快照内的数据文件标记为删除。仅当存在引用特定数据文件的类型为 DELETE 的清单条目时，数据文件才会被标记为删除。此标记可确保该文件不会被后续快照使用并可以安全删除。

假设上图中的所有 4 个快照都即将过期。过期流程如下：

* 它首先删除所有标记的数据文件，并记录任何更改的存储桶。
* 然后它会删除所有更改日志文件和关联的清单。
* 最后，它删除快照本身并写入最早的提示文件。

如果删除过程后有任何目录留空，它们也将被删除。

假设创建了另一个快照 snapshot-5 并触发了快照过期。 snapshot-1 到 snapshot-4 被删除。为简单起见，我们将只关注以前快照中的文件，快照过期后的最终布局如下所示：

图示

描述已自动生成

结果，分区20230503至20230510被物理删除。

### 2.10.6 Flink 流式写入

用 CDC 摄取的示例来说明 Flink Stream Write。本节将讨论更改数据的捕获和写入 Paimon，以及异步Compaction和快照提交和过期背后的机制。

CDC 摄取工作流程以及所涉及的每个组件所扮演的独特角色：

图示

描述已自动生成

（1）MySQL CDC Source统一读取快照和增量数据，分别由SnapshotReader读取快照数据和BinlogReader读取增量数据。

（2）Paimon Sink将数据写入桶级别的Paimon表中。其中的CompactManager将异步触发Compaction。

（3）Committer Operator 是一个单例，负责提交和过期快照。

端到端数据流：

图示

描述已自动生成

MySQL Cdc Source读取快照和增量数据，并在规范化后将它们发送到下游：

图示

描述已自动生成

Paimon Sink 首先将新记录缓冲在基于堆的 LSM 树中，并在内存缓冲区满时将它们刷新到磁盘。请注意，写入的每个数据文件都是Sorted Run。此时，还没有创建清单文件和快照。在 Flink 检查点发生之前，Paimon Sink 将刷新所有缓冲记录并向下游发送可提交消息，该消息在检查点期间由 Committer Operator 读取并提交：

图示

描述已自动生成

在检查点期间，Committer Operator 将创建一个新快照并将其与清单列表关联起来，以便该快照包含有关表中所有数据文件的信息：

图示

描述已自动生成

稍后可能会发生异步Compaction，CompactManager 生成的提交表包含有关先前文件和合并文件的信息，以便 Committer Operator 可以构造相应的清单条目。在这种情况下，Committer Operator 可能会在 Flink 检查点期间生成两个快照：

* 一个用于写入数据（Append 类型的快照），
* 另一个用于compact（Compact 类型的快照）。

如果在检查点间隔期间没有写入数据文件，则只会创建 Compact 类型的快照。 Committer Operator 将检查快照是否过期并执行标记数据文件的物理删除。

# 第3章 集成 Hive 引擎

前面与Flink集成时，通过使用 paimon Hive Catalog，可以从 Flink 创建、删除、查询和插入到 paimon 表中。这些操作直接影响相应的Hive元存储。以这种方式创建的表也可以直接从 Hive 访问。

更进一步的与 Hive 集成，可以使用 Hive SQL创建、查询Paimon表。

## 3.1 环境准备

Paimon 目前支持 Hive 3.1、2.3、2.2、2.1 和 2.1-cdh-6.3。支持 Hive Read 的 MR 和 Tez 执行引擎，以及 Hive Write 的 MR 执行引擎（beeline也不支持hive write）。

在Hive根目录下创建auxlib文件夹，将paimon-hive-connector-0.5-SNAPSHOT.jar复制到auxlib中（不推荐用add jar，MR 引擎运行 join 语句会报异常）：

下载地址：<https://repository.apache.org/snapshots/org/apache/paimon/paimon-hive-connector-3.1/0.5-SNAPSHOT/>

mkdir /opt/module/hive/auxlib

cp paimon-hive-connector-3.1-0.5-20230703.002437-65.jar /opt/module/hive/auxlib

## 3.2 访问已有的Paimon表

USE test;

SHOW TABLES;

SELECT \* FROM ws\_t;

INSERT INTO test\_table VALUES (9,9,9);

## 3.3 创建Paimon表

SET hive.metastore.warehouse.dir=hdfs://hadoop102:8020/paimon/hive;

CREATE TABLE test\_h(

a INT COMMENT 'The a field',

b STRING COMMENT 'The b field'

)

STORED BY 'org.apache.paimon.hive.PaimonStorageHandler'

## 3.4 通过外部表访问Paimon表

要访问现有的 paimon 表，还可以将它们注册为 Hive 中的外部表,不需要指定任何列或表属性，只需要指定路径。

CREATE EXTERNAL TABLE test.hive\_ex

STORED BY 'org.apache.paimon.hive.PaimonStorageHandler'

LOCATION 'hdfs://hadoop102:8020/paimon/hive/test.db/ws\_t';

--或将路径写在表属性中：

CREATE EXTERNAL TABLE hive\_ex

STORED BY 'org.apache.paimon.hive.PaimonStorageHandler'

TBLPROPERTIES (

'paimon\_location' ='hdfs://hadoop102:8020/paimon/hive/test.db/ws\_t'

);

操作外部表：

SELECT \* FROM hive\_ex;

INSERT INTO hive\_ex VALUES (8,8,8);

# 第4章 集成 Spark 引擎

## 4.1 环境准备

Paimon 目前支持 Spark 3.4、3.3、3.2 和 3.1。课程使用的Spark版本是3.3.1。

**1）上传并解压Spark安装包**

tar -zxvf spark-3.3.1-bin-hadoop3.tgz -C /opt/module/

mv /opt/module/spark-3.3.1-bin-hadoop3 /opt/module/spark-3.3.1

**2）配置环境变量**

sudo vim /etc/profile.d/my\_env.sh

export SPARK\_HOME=/opt/module/spark-3.3.1

export PATH=$PATH:$SPARK\_HOME/bin

source /etc/profile.d/my\_env.sh

**3）拷贝paimon的jar包到Spark的jars目录**

拷贝jar报到spark的jars目录（也可以运行时 --jars）

下载地址：<https://repository.apache.org/snapshots/org/apache/paimon/paimon-spark-3.3/0.5-SNAPSHOT/>

cp paimon-spark-3.3-0.5-20230703.002437-65.jar /opt/module/spark/jars

## 4.2 Catalog

启动spark-sql时，指定Catalog。切换到catalog后，Spark现有的表将无法直接访问，可以使用spark\_catalog.${database\_name}.${table\_name}来访问Spark表。

注册catalog可以启动时指定，也可以配置在spark-defaults.conf中

### 4.2.1 文件系统

spark-sql \

--conf spark.sql.catalog.fs=org.apache.paimon.spark.SparkCatalog \

--conf spark.sql.catalog.fs.warehouse=hdfs://hadoop102:8020/spark/paimon/fs

其中，参数前缀为：spark.sql.catalog.<catalog名称>

USE fs.default;

### 4.2.2 Hive

**1）启动hive的metastore服务**

nohup hive --service metastore &

**2）启动时注册Catalog**

spark-sql \

--conf spark.sql.catalog.hive=org.apache.paimon.spark.SparkCatalog \

--conf spark.sql.catalog.hive.warehouse=hdfs://hadoop102:8020/spark/paimon/hive \

--conf spark.sql.catalog.hive.metastore=hive \

--conf spark.sql.catalog.hive.uri=thrift://hadoop102:9083

切换到该catalog下的default数据库：

USE hive.default;

**3）禁用 Hive ACID（Hive3）**

hive.strict.managed.tables=false

hive.create.as.insert.only=false

metastore.create.as.acid=false

使用hive Catalog通过alter table更改不兼容的列类型时，参见 HIVE-17832。需要配置

hive.metastore.disallow.inknown.col.type.changes=false

## 4.3 DDL

### 4.3.1 建表

4.3.1.1 管理表

在 Paimon Catalog中创建的表就是Paimon的管理表，由Catalog管理。当表从Catalog中删除时，其表文件也将被删除，类似于Hive的内部表。

**1）创建表**

CREATE TABLE tests (

user\_id BIGINT,

item\_id BIGINT,

behavior STRING,

dt STRING,

hh STRING

) TBLPROPERTIES (

'primary-key' = 'dt,hh,user\_id'

);

**2）创建分区表**

CREATE TABLE tests\_p (

user\_id BIGINT,

item\_id BIGINT,

behavior STRING,

dt STRING,

hh STRING

) PARTITIONED BY (dt, hh)

TBLPROPERTIES (

'primary-key' = 'dt,hh,user\_id'

);

通过配置partition.expiration-time，可以自动删除过期的分区。

如果定义了主键，则分区字段必须是主键的子集。

可以定义以下三类字段为分区字段：

* 创建时间（推荐）：创建时间通常是不可变的，因此您可以放心地将其视为分区字段并将其添加到主键中。
* 事件时间：事件时间是原表中的一个字段。对于CDC数据来说，比如从MySQL CDC同步的表或者Paimon生成的Changelogs，它们都是完整的CDC数据，包括UPDATE\_BEFORE记录，即使你声明了包含分区字段的主键，也能达到独特的效果。
* CDC op\_ts：不能定义为分区字段，无法知道之前的记录时间戳。

**3）Create Table As**

表可以通过查询的结果创建和填充，例如，我们有一个这样的sql： CREATE TABLE table\_b AS SELECT id, name FORM table\_a, 生成的表table\_b将相当于创建表并插入数据以下语句：CREATE TABLE table\_b(id INT, name STRING); INSERT INTO table\_b SELECT id, name FROM table\_a;

使用CREATE TABLE AS SELECT时我们可以指定主键或分区。

CREATE TABLE tests1(

user\_id BIGINT,

item\_id BIGINT

);

CREATE TABLE tests2 AS SELECT \* FROM tests1;

-- 指定分区

CREATE TABLE tests2\_p PARTITIONED BY (dt) AS SELECT \* FROM tests\_p;

-- 指定配置

CREATE TABLE tests3(

user\_id BIGINT,

item\_id BIGINT

) TBLPROPERTIES ('file.format' = 'orc');

CREATE TABLE tests3\_op TBLPROPERTIES ('file.format' = 'parquet') AS SELECT \* FROM tests3;

-- 指定主键

CREATE TABLE tests\_pk TBLPROPERTIES ('primary-key' = 'dt') AS SELECT \* FROM tests;

-- 指定主键和分区

CREATE TABLE tests\_all PARTITIONED BY (dt) TBLPROPERTIES ('primary-key' = 'dt,hh') AS SELECT \* FROM tests\_p;

**4）表属性**

用户可以指定表属性来启用Paimon的功能或提高Paimon的性能。有关此类属性的完整列表，请参阅配置<https://paimon.apache.org/docs/master/maintenance/configurations/>。

CREATE TABLE tbl(

user\_id BIGINT,

item\_id BIGINT,

behavior STRING,

dt STRING,

hh STRING

) PARTITIONED BY (dt, hh)

TBLPROPERTIES (

'primary-key' = 'dt,hh,user\_id',

'bucket' = '2',

'bucket-key' = 'user\_id'

);

4.3.1.2 外部表

外部表由Catalog记录但不管理。如果删除外部表，其表文件不会被删除，类似于Hive的外部表。

Paimon 外部表可以在任何Catalog中使用。如果您不想创建Paimon Catalog而只想读/写表，则可以考虑外部表。

Spark3仅支持通过Scala API创建外部表。以下 Scala 代码将位于 hdfs:///path/to/table 的表加载到 DataSet 中。

val dataset = spark.read.format("paimon").load("hdfs:///path/to/table")

### 4.3.2 修改表

4.3.2.1 修改表

**1）更改/添加表属性**

ALTER TABLE tests SET TBLPROPERTIES (

'write-buffer-size' = '256 MB'

);

**2）重命名表名称**

ALTER TABLE tests1 RENAME TO tests\_new;

**3）删除表属性**

ALTER TABLE tests UNSET TBLPROPERTIES ('write-buffer-size');

4.3.2.2 修改列

**1）添加新列**

ALTER TABLE tests ADD COLUMNS (c1 INT, c2 STRING);

**2）重命名列名称**

ALTER TABLE tests RENAME COLUMN c1 TO c0;

**3）删除列**

ALTER TABLE my\_table DROP COLUMNS(c0, c2);

**4）更改列的可为空性**

CREATE TABLE tests\_null(

id INT,

coupon\_info FLOAT NOT NULL

);

-- Spark只支持将not null改为 nullable

ALTER TABLE tests\_null ALTER COLUMN coupon\_info DROP NOT NULL;

**5）更改列注释**

ALTER TABLE tests ALTER COLUMN user\_id COMMENT 'user id'

**6）添加列位置**

ALTER TABLE tests ADD COLUMN a INT FIRST;

ALTER TABLE tests ADD COLUMN b INT AFTER a;

注意：这种操作在hive中是不允许的，使用hive catalog无法执行，需要关闭hive的参数限制：

vim /opt/module/hive/conf/hive-site.xml;

<property>

<name>hive.metastore.disallow.incompatible.col.type.changes</name>

<value>false</value>

</property>

重启hive metastore服务。

**7）更改列位置**

ALTER TABLE tests ALTER COLUMN b FIRST;

ALTER TABLE tests ALTER COLUMN a AFTER user\_id;

**8）更改列类型**

ALTER TABLE tests ALTER COLUMN a TYPE DOUBLE;

## 4.4. 插入数据

INSERT 语句向表中插入新行。插入的行可以由值表达式或查询结果指定，跟标准的sql语法一致。

INSERT INTO table\_identifier [ part\_spec ] [ column\_list ] { value\_expr | query }

* part\_spec

可选，指定分区的键值对列表，多个用逗号分隔。可以使用类型文字（例如，date'2019-01-02'）。

语法： PARTITION (分区列名称 = 分区列值 [ , … ] )

* column\_list

可选，指定以逗号分隔的字段列表。

语法：(col\_name1 [,column\_name2, …])

所有指定的列都应该存在于表中，并且不能相互重复。它包括除静态分区列之外的所有列。字段列表的大小应与 VALUES 子句或查询中的数据大小完全相同。

* value\_expr

指定要插入的值。可以插入显式指定的值或 NULL。必须使用逗号分隔子句中的每个值。可以指定多于一组的值来插入多行。

语法：VALUES ( { 值 | NULL } [ , … ] ) [ , ( … ) ]

注意：将 Nullable 字段写入 Not-null 字段

不能将另一个表的可为空列插入到一个表的非空列中。Spark可以使用nvl函数来处理，比如A表的key1是not null，B表的key2是nullable：

INSERT INTO A key1 SELECT nvl(key2, <non-null expression>) FROM B

案例：

INSERT INTO tests VALUES(1,1,'order','2023-07-01','1'), (2,2,'pay','2023-07-01','2');

INSERT INTO tests\_p SELECT \* from tests;

## 4.5. 查询数据

就像所有其他表一样，Paimon 表可以使用 SELECT 语句进行查询。

Paimon的批量读取返回表快照中的所有数据。默认情况下，批量读取返回最新快照。

### 4.5.1 时间旅行

可以在查询中使用 VERSION AS OF 和 TIMESTAMP AS OF 来进行时间旅行。

**1）读取指定id的快照**

SELECT \* FROM tests VERSION AS OF 1;

SELECT \* FROM tests VERSION AS OF 2;

**2）读取指定时间戳的快照**

-- 查看快照信息

SELECT \* FROM `tests&snapshots`;

SELECT \* FROM tests TIMESTAMP AS OF '2023-07-03 15:34:20.123';

-- 时间戳指定到秒（向上取整）

SELECT \* FROM tests TIMESTAMP AS OF 1688369660;

**3）读取指定标签**

SELECT \* FROM tests VERSION AS OF 'my-tag';

### 4.5.2 增量查询

读取开始快照（不包括）和结束快照之间的增量更改。例如，“3,5”表示快照 3 和快照 5 之间的更改：

spark.read()

.format("paimon")

.option("incremental-between", "3,5")

.load("path/to/table")

## 4.6 系统表

系统表包含有关每个表的元数据和信息，例如创建的快照和使用的选项。用户可以通过批量查询访问系统表。

### 4.6.1 快照表 Snapshots Table

通过snapshots表可以查询表的快照历史信息，包括快照中发生的记录数。Spark中使用需要反引号`表名$系统表名`。

SELECT \* FROM `tests$snapshots`;

通过查询快照表，可以了解该表的提交和过期信息以及数据的时间旅行。

### 4.6.2 模式表 Schemas Table

通过schemas表可以查询该表的历史schema。

SELECT \* FROM `tests$schemas`;

可以连接快照表和模式表以获取给定快照的字段。

SELECT s.snapshot\_id, t.schema\_id, t.fields

FROM `tests$snapshots` s JOIN `tests$schemas` t

ON s.schema\_id=t.schema\_id where s.snapshot\_id=3;

### 4.6.3 选项表 Options Table

可以通过选项表查询DDL中指定的表的选项信息。未显示的选项将是默认值。

SELECT \* FROM `tests$options`;

### 4.6.4 审计日志表 Audit log Table

如果需要审计表的changelog，可以使用audit\_log系统表。通过audit\_log表，获取表增量数据时可以获取rowkind列。您可以利用该栏目进行过滤等操作来完成审核。

rowkind 有四个值：

+I：插入操作。

-U：使用更新行的先前内容进行更新操作。

+U：使用更新行的新内容进行更新操作。

-D：删除操作。

SELECT \* FROM `tests$audit\_log`;

### 4.6.5 文件表 Files Table

可以查询特定快照表的文件。

-- 查询最新快照的文件

SELECT \* FROM `tests$files`;

### 4.6.6 标签表 Tags Table

通过tags表可以查询表的标签历史信息，包括基于哪些快照进行标签以及快照的一些历史信息。您还可以通过名称获取所有标签名称和时间旅行到特定标签的数据。

SELECT \* FROM `tests$tags`;